Le manuel de référence du langage METAQUOTES LANGUAGE 5

pour le terminal de client MetaTrader 5

ÉTUDIEZ le langage MQL5 et résoudrez toutes les tâches:

- La création des indicateurs personnels de l'analyse technique de n'importe quelle complexité
- L’autotrading – c’est l'automatisation du système commercial pour travailler sur les marchés financiers différents
- Le développement des outils analytiques à la base d’acquisitions mathématiques et des méthodes classiques
- L'écriture des systèmes informatiques commerciaux pour résoudre une large gamme des tâches – le commerce, le monitoring, des signaux

© 2000-2024, MetaQuotes Ltd

Teneur

Référence MQL5 71

1 Concepts de Base du Langage ... 73

Syntaxe ... 74
Commentaires ... 75
Identifiants ... 76
Mots Réservés ... 77
Types de Données ... 79
Types Integer ... 80
Types Char, Short, Int et Long ... 81
Constantes Caractères ... 85
Type Datetime ... 89
Type Color ... 90
Type Bool ... 91
Enumérations ... 92
Types Réels (double, float) ... 94
Nombre complexe (complex) ... 99
Type String ... 100
Structures, Classes et Interfaces ... 101
Tableau Dynamique d'Objets ... 129
Matrices et vecteurs ... 130
Conformation des types ... 138
Type Void et Constante NULL ... 143
Types des données ... 144
Pointeurs d'Objets ... 154
Références : Modificateur & et Mot-Clé this ... 158
Opérations et Expressions ... 160
Expressions ... 161
Opérations Arithmétiques ... 162
Opérations d'Assignement ... 163
Opérations de Relation ... 164
Opérations Booléennes ... 165
Opérations sur les Bits ... 167
Autres Opérations ... 170
Règles de Précédence ... 174
Opérateurs ... 176
Opérateurs Composés ... 177
Opérateur d'Expression ... 178
Opérateur Return ... 179
Opérateur Conditionnel if-else ... 180
Opérateur Ternaire ?: ... 181
Opérateur Switch ... 183
Opérateur de Boucle while ... 185
Opérateur de Boucle for ... 186
Opérateur de Boucle do while ... 188
Opérateur Break ... 189
Opérateur Continue ... 190
Opérateur de Création d'Objet new ... 191
Opérateur de Suppression d'Objet delete ... 193
Fonctions ... 194
Appel de Fonction ... 196
La transmission des paramètres ... 198
Surcharge de Fonction ... 201
Surcharge d'Opération ... 204

© 2000-2024, MetaQuotes Ltd.

Description des Fonctions Externes ... 218
Export de Fonctions ... 220
Fonctions de Gestion d'Evènement ... 221
Variables ... 233
Variables Locales ... 237
Paramètres Formels ... 239
Variables Statiques ... 241
Variables Globales ... 243
Variables d'Entrée ... 244
Variables Externes ... 251
Initialisation des Variables ... 252
Portée de la Visibilité et Cycle de Vie des Variables ... 254
Créer et Supprimer des Objets ... 256
Préprocesseur ... 259
Macro de substitution (#define) ... 261
Propriétés du Programme (#property) ... 264
Inclure des Fichiers (#include) ... 270
Importer des Fonctions (#import) ... 271
Compilation Conditionnelle (#ifdef, #ifndef, #else, #endif) ... 274
Programmation Orientée Objet ... 276
Encapsulation et Extension des Types ... 278
Héritage ... 281
Polymorphisme ... 286
Surcharge ... 290
Fonctions virtuelles ... 291
Membres Statiques de Classe ... 295
Templates de fonctions ... 299
Templates de classes ... 304
Abstract Classes ... 309
Eespace de noms ... 311
2 Constantes, Enumérations et Structures ... 315
Constantes de Graphique ... 316
Types des Evénements du Graphique ... 317
Périodes du Graphique ... 325
Propriétés du Graphique ... 327
Constantes de Positionnement ... 335
Représentation du Graphique ... 336
Exemples d'Utilisation du Graphique ... 338
Constantes d'Objets ... 396
Types d'Objets ... 397
OBJ_VLINE ... 399
OBJ_HLINE ... 404
OBJ_TREND ... 409
OBJ_TRENDBYANGLE ... 416
OBJ_CYCLES ... 422
OBJ_ARROWED_LINE ... 428
OBJ_CHANNEL ... 434
OBJ_STDDEVCHANNEL ... 441
OBJ_REGRESSION ... 448
OBJ_PITCHFORK ... 454
OBJ_GANNLINE ... 462
OBJ_GANNFAN ... 469
OBJ_GANNGRID ... 476
OBJ_FIBO ... 483
OBJ_FIBOTIMES ... 490
OBJ_FIBOFAN ... 497
OBJ_FIBOARC ... 504
OBJ_FIBOCHANNEL ... 511

© 2000-2024, MetaQuotes Ltd.

OBJ_EXPANSION ... 519
OBJ_ELLIOTWAVE5 ... 527
OBJ_ELLIOTWAVE3 ... 535
OBJ_RECTANGLE ... 542
OBJ_TRIANGLE ... 548
OBJ_ELLIPSE ... 555
OBJ_ARROW_THUMB_UP ... 562
OBJ_ARROW_THUMB_DOWN ... 568
OBJ_ARROW_UP ... 574
OBJ_ARROW_DOWN ... 580
OBJ_ARROW_STOP ... 586
OBJ_ARROW_CHECK ... 592
OBJ_ARROW_LEFT_PRICE ... 598
OBJ_ARROW_RIGHT_PRICE ... 603
OBJ_ARROW_BUY ... 608
OBJ_ARROW_SELL ... 613
OBJ_ARROW ... 618
OBJ_TEXT ... 624
OBJ_LABEL ... 630
OBJ_BUTTON ... 638
OBJ_CHART ... 645
OBJ_BITMAP ... 652
OBJ_BITMAP_LABEL ... 659
OBJ_EDIT ... 666
OBJ_EVENT ... 673
OBJ_RECTANGLE_LABEL ... 678
Propriétés d'Objets ... 684
Méthodes de liaison d'un Objet ... 717
Coin du Graphique ... 722
Visibilité des Objets ... 725
Niveaux des Vagues d'Elliott ... 728
Objets de Gann ... 729
Couleurs du Web ... 731
Wingdings ... 733
Constantes d'Indicateur ... 734
Constantes de Prix ... 735
Méthodes de Lissage ... 738
Lignes d'Indicateurs ... 739
Styles de Dessin ... 741
Propriétés des Indicateurs Personnels ... 747
Types d'Indicateurs ... 754
Identifiants des Types de Données ... 756
Etat de l'Environnement de Travail ... 757
Propriétés du Terminal Client ... 758
Propriétés des Programmes MQL5 ... 764
Propriétés du Symbole ... 768
Propriétés du Compte ... 897
Statistiques de Test ... 908
Constantes de Trading ... 913
Propriétés de l'Historique ... 914
Propriétés d'un Ordre ... 915
Propriétés d'une Position ... 934
Propriétés d'un Deal ... 939
Types d'Opérations de Trading ... 944
Types de Transactions de Trading ... 956
Ordres de Trading en DOM ... 959
Propriétés des Signaux ... 960
Constantes Nommées ... 962

© 2000-2024, MetaQuotes Ltd.

Macros de Substitutions Prédéfinies ... 963
Constantes Mathématiques ... 969
Constantes de Type Numerique ... 971
Codes des Raisons de Dé-initialisation ... 974
Vérifier un Pointeur d'Objet ... 976
Autres Constantes ... 977
Structures de Données ... 981
Structure du Type Date ... 982
Structure des Paramètres d'Entrée ... 983
Structure des Données Historiques ... 984
Structure DOM ... 985
Structure de la Requête de Trading ... 986
Structure des Résultats de la Vérification d'une Requête ... 999
Structure des Résultat d'une Requête de Trading ... 1000
Structure d'une Transaction de Trading ... 1004
Structure des Prix Courants ... 1012
Structures du Calendrier Economique ... 1014
Codes d'Erreurs et Avertissements ... 1023
Codes de Retour du Serveur de Trading ... 1024
Avertissements du Compilateur ... 1028
Erreurs de Compilation ... 1031
Erreurs d'Exécution ... 1042
Constantes d'Entrée/Sortie ... 1056
Flags d'Ouverture des Fichiers ... 1057
Propriétés de Fichiers ... 1059
Position dans le Fichier ... 1060
Utilisation d'une Page de Code ... 1061
MessageBox ... 1062
3 Programmes MQL5 ... 1064
Exécution du Programme ... 1065
Autorisation de trading ... 1072
Evénements du Terminal Client ... 1076
Ressources ... 1080
Appel aux Fonctions Importées ... 1092
Erreurs d'Exécution ... 1094
Test des Stratégies de Trading ... 1095
4 Variables Prédéfinies ... 1122
_AppliedTo ... 1123
_Digits ... 1125
_Point ... 1126
_LastError ... 1127
_Period ... 1128
_RandomSeed ... 1129
_StopFlag ... 1130
_Symbol ... 1131
_UninitReason ... 1132
_IsX64 ... 1133
5 Fonctions Communes ... 1134
Alert ... 1136
CheckPointer ... 1137
Comment ... 1139
CryptEncode ... 1141
CryptDecode ... 1143
DebugBreak ... 1144
ExpertRemove ... 1145
GetPointer ... 1147
GetTickCount ... 1150

© 2000-2024, MetaQuotes Ltd.

GetTickCount64 ... 1151
GetMicrosecondCount ... 1152
MessageBox ... 1154
PeriodSeconds ... 1155
PlaySound ... 1156
Print ... 1157
PrintFormat ... 1159
ResetLastError ... 1165
ResourceCreate ... 1166
ResourceFree ... 1168
ResourceReadImage ... 1169
ResourceSave ... 1170
SetReturnError ... 1171
SetUserError ... 1172
Sleep ... 1173
TerminalClose ... 1174
TesterHideIndicators ... 1176
TesterStatistics ... 1178
TesterStop ... 1179
TesterDeposit ... 1180
TesterWithdrawal ... 1181
TranslateKey ... 1182
ZeroMemory ... 1183

6 Fonctions sur les Tableaux ... 1184

ArrayBsearch ... 1185
ArrayCopy ... 1189
ArrayCompare ... 1194
ArrayFree ... 1195
ArrayGetAsSeries ... 1204
ArrayInitialize ... 1207
ArrayFill ... 1209
ArrayIsDynamic ... 1211
ArrayIsSeries ... 1213
ArrayMaximum ... 1215
ArrayMinimum ... 1226
ArrayPrint ... 1237
ArrayRange ... 1240
ArrayResize ... 1241
ArrayInsert ... 1244
ArrayRemove ... 1247
ArrayReverse ... 1249
ArraySetAsSeries ... 1251
ArraySize ... 1254
ArraySort ... 1256
ArraySwap ... 1261

7 Méthodes sur les Matrices et les Vecteurs ... 1263

Matrix and Vector Types ... 1271
Enumerations ... 1272
Initialization ... 1277
Assign ... 1280
CopyIndicatorBuffer ... 1282
CopyRates ... 1284
CopyTicks ... 1288
CopyTicksRange ... 1291
Eye ... 1293
Identity ... 1295
Ones ... 1297
Zeros ... 1298

© 2000-2024, MetaQuotes Ltd.

Full ... 1299
Tri ... 1300
Init ... 1302
Fill ... 1304
Manipulations ... 1305
HasNan ... 1307
Transpose ... 1308
TriL ... 1310
TriU ... 1311
Diag ... 1312
Row ... 1314
Col ... 1316
Copy ... 1318
Compare ... 1320
CompareByDigits ... 1322
Flat ... 1324
Clip ... 1326
Reshape ... 1327
Resize ... 1329
Set ... 1331
SwapRows ... 1333
SwapCols ... 1334
Split ... 1335
Hsplit ... 1337
Vsplit ... 1339
ArgSort ... 1341
Sort ... 1342
Operations ... 1344
Mathematical operations ... 1346
Mathematical functions ... 1347
Products ... 1348
MatMul ... 1349
GeMM ... 1354
Power ... 1358
Dot ... 1361
Kron ... 1363
Inner ... 1365
Outer ... 1367
CorrCoef ... 1370
Cov ... 1374
Correlate ... 1377
Convolve ... 1380
Transformations ... 1383
Cholesky ... 1384
Eig ... 1385
EigVals ... 1388
LU ... 1389
LUP ... 1391
QR ... 1393
SVD ... 1395
Statistics ... 1398
ArgMax ... 1399
ArgMin ... 1400
Max ... 1401
Min ... 1402
Ptp ... 1403
Sum ... 1404
Prod ... 1405

© 2000-2024, MetaQuotes Ltd.

CumSum ... 1407
CumProd ... 1409
Percentile ... 1411
Quantile ... 1413
Median ... 1415
Mean ... 1417
Average ... 1418
Std ... 1420
Var ... 1422
LinearRegression ... 1424
Features ... 1427
Rows ... 1428
Cols ... 1429
Size ... 1430
Norm ... 1431
Cond ... 1434
Det ... 1437
SLogDet ... 1439
Rank ... 1440
Trace ... 1443
Spectrum ... 1444
Solutions ... 1445
Solve ... 1446
LstSq ... 1447
Inv ... 1448
PInv ... 1450
Machine learning ... 1451
Activation ... 1456
Derivative ... 1460
Loss ... 1462
LossGradient ... 1464
RegressionMetric ... 1466
ConfusionMatrix ... 1468
ConfusionMatrixMultilabel ... 1470
ClassificationMetric ... 1472
ClassificationScore ... 1475
PrecisionRecall ... 1479
ReceiverOperatingCharacteristic ... 1483
8 Fonctions de Conversion ... 1487
CharToString ... 1489
CharArrayToString ... 1490
CharArrayToStruct ... 1491
StructToCharArray ... 1492
ColorToARGB ... 1493
ColorToString ... 1495
DoubleToString ... 1496
EnumToString ... 1497
IntegerToString ... 1499
ShortToString ... 1500
ShortArrayToString ... 1501
TimeToString ... 1502
NormalizeDouble ... 1503
StringToCharArray ... 1505
StringToColor ... 1506
StringToDouble ... 1507
StringToInteger ... 1508
StringToShortArray ... 1509
StringToTime ... 1510

© 2000-2024, MetaQuotes Ltd.

StringFormat 1511
9 Fonctions Mathématiques 1515
MathAbs 1517
MathArccos 1518
MathArcsin 1519
MathArctan 1520
MathArctan2 1521
MathClassify 1522
MathCeil 1524
MathCos 1525
MathExp 1526
MathFloor 1527
MathLog 1528
MathLog10 1529
MathMax 1530
MathMin 1531
MathMod 1532
MathPow 1533
MathRand 1534
MathRound 1535
MathSin 1536
MathSqrt 1537
MathSrand 1538
MathTan 1541
MathIsValidNumber 1542
MathExpm1 1543
MathLog1p 1544
MathArccosh 1545
MathArcsinh 1546
MathArctanh 1547
MathCosh 1548
MathSinh 1549
MathTanh 1550
MathSwap 1551
10 Fonctions sur les Chaînes de Caractères 1552
StringAdd 1553
StringBufferLen 1555
StringCompare 1556
StringConcatenate 1558
StringFill 1559
StringFind 1560
StringGetCharacter 1561
StringInit 1562
StringLen 1563
StringSetLength 1564
StringReplace 1565
StringReserve 1566
StringSetCharacter 1568
StringSplit 1570
StringSubstr 1571
StringToLower 1572
StringToUpper 1573
StringTrimLeft 1574
StringTrimRight 1575
11 Date et Heure 1576
TimeCurrent 1577
TimeTradeServer 1578

© 2000-2024, MetaQuotes Ltd.

TimeLocal 1579
TimeGMT 1580
TimeDaylightSavings 1581
TimeGMTOffset 1582
TimeToStruct 1583
StructToTime 1584
12 Information du Compte 1585
AccountInfoDouble 1586
AccountInfoInteger 1587
AccountInfoString 1589
13 Vérification 1590
GetLastError 1591
IsStopped 1592
UninitializeReason 1593
TerminalInfoInteger 1594
TerminalInfoDouble 1595
TerminalInfoString 1596
MQLInfoInteger 1597
MQLInfoString 1598
Symbol 1599
Period 1600
Digits 1601
Point 1602
14 Gestion d'Evènements 1603
OnStart 1605
OnInit 1608
OnDeinit 1611
OnTick 1614
OnCalculate 1620
OnTimer 1624
OnTrade 1627
OnTradeTransaction 1632
OnBookEvent 1638
OnChartEvent 1641
OnTester 1648
OnTesterInit 1655
OnTesterDeinit 1662
OnTesterPass 1663
15 Information du Marché 1664
SymbolsTotal 1665
SymbolExist 1666
SymbolName 1667
SymbolSelect 1668
SymbolIsSynchronized 1669
SymbolInfoDouble 1670
SymbolInfoInteger 1672
SymbolInfoString 1674
SymbolInfoMarginRate 1675
SymbolInfoTick 1676
SymbolInfoSessionQuote 1677
SymbolInfoSessionTrade 1678
MarketBookAdd 1679
MarketBookRelease 1680
MarketBookGet 1681
16 Calendrier Economique 1682
CalendarCountryById 1683

© 2000-2024, MetaQuotes Ltd.

CalendarEventById ... 1685
CalendarValueById ... 1688
CalendarCountries ... 1691
CalendarEventByCountry ... 1693
CalendarEventByCurrency ... 1695
CalendarValueHistoryByEvent ... 1697
CalendarValueHistory ... 1700
CalendarValueLastByEvent ... 1703
CalendarValueLast ... 1708

17 Accès aux Séries de Données et aux Indicateurs ... 1713

Direction d'Indexation dans les Tableaux, Buffers et Séries de Données ... 1718
Organisation de l'Accès aux Données ... 1722
SeriesInfoInteger ... 1732
Bars ... 1734
BarsCalculated ... 1737
IndicatorCreate ... 1739
IndicatorParameters ... 1742
IndicatorRelease ... 1744
CopyBuffer ... 1746
CopyRates ... 1751
CopySeries ... 1755
CopyTime ... 1759
CopyOpen ... 1762
CopyHigh ... 1765
CopyLow ... 1769
CopyClose ... 1772
CopyTickVolume ... 1775
CopyRealVolume ... 1779
CopySpread ... 1782
CopyTicks ... 1786
CopyTicksRange ... 1792
iBars ... 1794
iBarShift ... 1795
iClose ... 1798
iHigh ... 1800
iHighest ... 1802
iLow ... 1803
iLowest ... 1805
iOpen ... 1806
iTime ... 1808
iTickVolume ... 1810
iRealVolume ... 1812
iVolume ... 1814
iSpread ... 1816

18 Symboles personnalisés ... 1818

CustomSymbolCreate ... 1820
CustomSymbolDelete ... 1822
CustomSymbolSetInteger ... 1823
CustomSymbolSetDouble ... 1824
CustomSymbolSetString ... 1825
CustomSymbolSetMarginRate ... 1826
CustomSymbolSetSessionQuote ... 1827
CustomSymbolSetSessionTrade ... 1828
CustomRatesDelete ... 1829
CustomRatesReplace ... 1830
CustomRatesUpdate ... 1831
CustomTicksAdd ... 1832
CustomTicksDelete ... 1834

© 2000-2024, MetaQuotes Ltd.

CustomTicksReplace ... 1835
CustomBookAdd ... 1837
19 Opérations sur le Graphique ... 1840
ChartApplyTemplate ... 1842
ChartSaveTemplate ... 1845
ChartWindowFind ... 1850
ChartTimePriceToXY ... 1852
ChartXYToTimePrice ... 1853
ChartOpen ... 1855
ChartFirst ... 1856
ChartNext ... 1857
ChartClose ... 1858
ChartSymbol ... 1859
ChartPeriod ... 1860
ChartRedraw ... 1861
ChartSetDouble ... 1862
ChartSetInteger ... 1863
ChartSetString ... 1865
ChartGetDouble ... 1867
ChartGetInteger ... 1869
ChartGetString ... 1871
ChartNavigate ... 1873
ChartID ... 1876
ChartIndicatorAdd ... 1877
ChartIndicatorDelete ... 1880
ChartIndicatorGet ... 1883
ChartIndicatorName ... 1885
ChartIndicatorsTotal ... 1886
ChartWindowOnDropped ... 1887
ChartPriceOnDropped ... 1888
ChartTimeOnDropped ... 1889
ChartXOnDropped ... 1890
ChartYOnDropped ... 1891
ChartSetSymbolPeriod ... 1892
ChartScreenShot ... 1893
20 Fonctions de Trading ... 1896
OrderCalcMargin ... 1898
OrderCalcProfit ... 1899
OrderCheck ... 1900
OrderSend ... 1901
OrderSendAsync ... 1906
PositionsTotal ... 1917
PositionGetSymbol ... 1918
PositionSelect ... 1919
PositionSelectByTicket ... 1920
PositionGetDouble ... 1921
PositionGetInteger ... 1922
PositionGetString ... 1924
PositionGetTicket ... 1925
OrdersTotal ... 1926
OrderGetTicket ... 1927
OrderSelect ... 1929
OrderGetDouble ... 1930
OrderGetInteger ... 1931
OrderGetString ... 1932
HistorySelect ... 1933
HistorySelectByPosition ... 1935
HistoryOrderSelect ... 1936

© 2000-2024, MetaQuotes Ltd.

HistoryOrdersTotal ... 1937
HistoryOrderGetTicket ... 1938
HistoryOrderGetDouble ... 1940
HistoryOrderGetInteger ... 1941
HistoryOrderGetString ... 1944
HistoryDealSelect ... 1945
HistoryDealsTotal ... 1946
HistoryDealGetTicket ... 1947
HistoryDealGetDouble ... 1949
HistoryDealGetInteger ... 1950
HistoryDealGetString ... 1953

21 Signaux de Trading ... 1954

SignalBaseGetDouble ... 1955
SignalBaseGetInteger ... 1956
SignalBaseGetString ... 1957
SignalBaseSelect ... 1958
SignalBaseTotal ... 1959
SignalInfoGetDouble ... 1960
SignalInfoGetInteger ... 1961
SignalInfoGetString ... 1962
SignalInfoSetDouble ... 1963
SignalInfoSetInteger ... 1964
SignalSubscribe ... 1965
SignalUnSubscribe ... 1966

22 Fonctions réseau ... 1967

SocketCreate ... 1969
SocketClose ... 1972
SocketConnect ... 1975
SocketIsConnected ... 1978
SocketIsReadable ... 1979
SocketIsWritable ... 1982
SocketTimeouts ... 1983
SocketRead ... 1984
SocketSend ... 1987
SocketTlsHandshake ... 1990
SocketTlsCertificate ... 1991
SocketTlsRead ... 1994
SocketTlsReadAvailable ... 1997
SocketTlsSend ... 1998
WebRequest ... 1999
SendFTP ... 2002
SendMail ... 2003
SendNotification ... 2004

23 Variables Globales du Terminal ... 2005

GlobalVariableCheck ... 2006
GlobalVariableTime ... 2007
GlobalVariableDel ... 2008
GlobalVariableGet ... 2009
GlobalVariableName ... 2010
GlobalVariableSet ... 2011
GlobalVariablesFlush ... 2012
GlobalVariableTemp ... 2013
GlobalVariableSetOnCondition ... 2014
GlobalVariablesDeleteAll ... 2015
GlobalVariablesTotal ... 2016

24 Fonctions sur les Fichiers ... 2017

FileSelectDialog ... 2020

© 2000-2024, MetaQuotes Ltd.

FileFindFirst 2023
FileFindNext 2025
FileFindClose 2027
FileIsExist 2028
FileOpen 2031
FileClose 2034
FileCopy 2035
FileDelete 2038
FileMove 2041
FileFlush 2043
FileGetInteger 2045
FileIsEnding 2048
FileIsLineEnding 2049
FileReadArray 2055
FileReadBool 2057
FileReadDatetime 2061
FileReadDouble 2065
FileReadFloat 2068
FileReadInteger 2072
FileReadLong 2076
FileReadNumber 2079
FileReadString 2085
FileReadStruct 2087
FileSeek 2091
FileSize 2094
FileTell 2097
FileWrite 2100
FileWriteArray 2103
FileWriteDouble 2106
FileWriteFloat 2109
FileWriteInteger 2112
FileWriteLong 2115
FileWriteString 2118
FileWriteStruct 2121
FileLoad 2124
FileSave 2126
FolderCreate 2128
FolderDelete 2131
FolderClean 2134

25 Indicateurs Personnalisés 2137

Styles d'Indicateurs dans les Exemples 2141
DRAW_NONE 2154
DRAW_LINE 2157
DRAW_SECTION 2161
DRAW_HISTOGRAM 2165
DRAW_HISTOGRAM2 2169
DRAW_ARROW 2173
DRAW_ZIGZAG 2178
DRAW_FILLING 2183
DRAW_BARS 2188
DRAW_CANDLES 2194
DRAW_COLOR_LINE 2201
DRAW_COLOR_SECTION 2206
DRAW_COLOR_HISTOGRAM 2212
DRAW_COLOR_HISTOGRAM2 2217
DRAW_COLOR_ARROW 2222
DRAW_COLOR_ZIGZAG 2228
DRAW_COLOR_BARS 2233

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_CANDLES ... 2240
Lien entre les Propriétés d'un Indicateur et les Fonctions ... 2247
SetIndexBuffer ... 2250
IndicatorSetDouble ... 2253
IndicatorSetInteger ... 2257
IndicatorSetString ... 2261
PlotIndexSetDouble ... 2264
PlotIndexSetInteger ... 2265
PlotIndexSetString ... 2269
PlotIndexGetInteger ... 2270
26 Fonctions sur les Objets ... 2273
ObjectCreate ... 2275
ObjectName ... 2279
ObjectDelete ... 2280
ObjectsDeleteAll ... 2281
ObjectFind ... 2282
ObjectGetTimeByValue ... 2283
ObjectGetValueByTime ... 2284
ObjectMove ... 2285
ObjectsTotal ... 2286
ObjectSetDouble ... 2287
ObjectSetInteger ... 2291
ObjectSetString ... 2295
ObjectGetDouble ... 2297
ObjectGetInteger ... 2299
ObjectGetString ... 2301
TextSetFont ... 2303
TextOut ... 2305
TextGetSize ... 2309
27 Indicateurs Techniques ... 2310
iAC ... 2313
iAD ... 2318
iADX ... 2323
iADXWilder ... 2328
iAlligator ... 2333
iAMA ... 2340
iAO ... 2345
iATR ... 2350
iBearsPower ... 2355
iBands ... 2360
iBullsPower ... 2366
iCCI ... 2371
iChaikin ... 2376
iCustom ... 2381
iDEMA ... 2384
iDeMarker ... 2389
iEnvelopes ... 2394
iForce ... 2400
iFractals ... 2405
iFrAMA ... 2410
iGator ... 2415
iIchimoku ... 2422
iBWMFI ... 2429
iMomentum ... 2434
iMFI ... 2439
iMA ... 2444
iOsMA ... 2449
iMACD ... 2454

© 2000-2024, MetaQuotes Ltd.

iOBV 2460
iSAR 2465
iRSI 2470
iRVI 2475
iStdDev 2480
iStochastic 2485
iTEMA 2491
iTriX 2496
iWPR 2501
iVIDyA 2506
iVolumes 2511

28 Utilisation des Résultats d'Optimisation 2516

FrameFirst 2518
FrameFilter 2519
FrameNext 2520
FrameInputs 2521
FrameAdd 2522
ParameterGetRange 2523
ParameterSetRange 2527

29 Utilisation des Evènements 2529

EventSetMillisecondTimer 2530
EventSetTimer 2531
EventKillTimer 2532
EventChartCustom 2533

30 Utilisation d'OpenCL 2539

CLHandleType 2541
CLGetInfoInteger 2542
CLGetInfoString 2545
CLContextCreate 2548
CLContextFree 2549
CLGetDeviceInfo 2550
CLProgramCreate 2555
CLProgramFree 2559
CLKernelCreate 2560
CLKernelFree 2561
CLSetKernelArg 2562
CLSetKernelArgMem 2563
CLSetKernelArgMemLocal 2564
CLBufferCreate 2565
CLBufferFree 2566
CLBufferWrite 2567
CLBufferRead 2572
CLExecute 2576
CLExecutionStatus 2578

31 Utilisation de bases de données 2579

DatabaseOpen 2583
DatabaseClose 2585
DatabaseImport 2586
DatabaseExport 2589
DatabasePrint 2595
DatabaseTableExists 2600
DatabaseExecute 2601
DatabasePrepare 2613
DatabaseReset 2622
DatabaseBind 2628
DatabaseBindArray 2633
DatabaseRead 2638

© 2000-2024, MetaQuotes Ltd.

DatabaseReadBind 2639
DatabaseFinalize 2643
DatabaseTransactionBegin 2644
DatabaseTransactionCommit 2649
DatabaseTransactionRollback 2650
DatabaseColumnsCount 2651
DatabaseColumnName 2652
DatabaseColumnType 2653
DatabaseColumnSize 2654
DatabaseColumnText 2655
DatabaseColumnInteger 2656
DatabaseColumnLong 2657
DatabaseColumnDouble 2658
DatabaseColumnBlob 2659

32 Utilisation de DirectX 2660

DXContextCreate 2662
DXContextSetSize 2663
DXContextGetSize 2664
DXContextClearColors 2665
DXContextClearDepth 2666
DXContextGetColors 2667
DXContextGetDepth 2668
DXBufferCreate 2669
DXTextureCreate 2670
DXInputCreate 2676
DXInputSet 2677
DXShaderCreate 2678
DXShaderSetLayout 2679
DXShaderInputsSet 2680
DXShaderTexturesSet 2681
DXDraw 2682
DXDrawIndexed 2683
DXPrimiveTopologySet 2684
DXBufferSet 2685
DXShaderSet 2686
DXHandleType 2687
DXRelease 2688

33 MetaTrader pour Python 2689

initialize 2695
login 2697
shutdown 2700
version 2701
last_error 2703
account_info 2705
terminal_info 2708
symbols_total 2711
symbols_get 2712
symbol_info 2715
symbol_info_tick 2719
symbol_select 2721
market_book_add 2725
market_book_get 2726
market_book_release 2729
copy_rates_from 2730
copy_rates_from_pos 2734
copy_rates_range 2737
copy_ticks_from 2740
copy_ticks_range 2744

© 2000-2024, MetaQuotes Ltd.

orders_total 2747
orders_get 2748
order_calc_margin 2751
order_calc_profit 2754
order_check 2757
order_send 2761
positions_total 2766
positions_get 2767
history_orders_total 2770
history_orders_get 2772
history_deals_total 2775
history_deals_get 2777
34 Modèles ONNX 2781
Support ONNX 2782
Conversion de Formats 2784
Conversion automatique des types 2785
Création d'un modèle 2789
Exécution d'un modèle 2797
Exécuter dans le Testeur de Stratégie 2803
OnnxCreate 2808
OnnxCreateFromBuffer 2809
OnnxRelease 2810
OnnxRun 2811
OnnxGetInputCount 2814
OnnxGetOutputCount 2815
OnnxGetInputName 2816
OnnxGetOutputName 2817
OnnxGetInputTypeInfo 2818
OnnxGetOutputTypeInfo 2819
OnnxSetInputShape 2820
OnnxSetOutputShape 2821
Structures de données 2822
35 Bibliothèque Standard 2825
Mathematiques 2826
Statistiques 2827
Caractéristiques Statistiques 2830
MathMean 2831
MathVariance 2832
MathSkewness 2833
MathKurtosis 2834
MathMoments 2835
MathMedian 2836
MathStandardDeviation 2837
MathAverageDeviation 2838
Distribution Normale 2839
MathProbabilityDensityNormal 2840
MathCumulativeDistributionNormal 2842
MathQuantileNormal 2844
MathRandomNormal 2846
MathMomentsNormal 2847
Distribution Normale 2848
MathProbabilityDensityLognormal 2849
MathCumulativeDistributionLognormal 2851
MathQuantileLognormal 2853
MathRandomLognormal 2855
MathMomentsLognormal 2856
Distribution beta 2857
MathProbabilityDensityBeta 2858

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionBeta ... 2860
MathQuantileBeta ... 2862
MathRandomBeta ... 2864
MathMomentsBeta ... 2865
Distribution bêta non centrale ... 2866
MathProbabilityDensityNoncentralBeta ... 2867
MathCumulativeDistributionNoncentralBeta ... 2869
MathQuantileNoncentralBeta ... 2871
MathRandomNoncentralBeta ... 2873
MathMomentsNoncentralBeta ... 2874
Distribution gamma ... 2875
MathProbabilityDensityGamma ... 2876
MathCumulativeDistributionGamma ... 2878
MathQuantileGamma ... 2880
MathRandomGamma ... 2882
MathMomentsGamma ... 2883
Distribution khi-deux ... 2884
MathProbabilityDensityChiSquare ... 2885
MathCumulativeDistributionChiSquare ... 2887
MathQuantileChiSquare ... 2889
MathRandomChiSquare ... 2891
MathMomentsChiSquare ... 2892
Distribution khi-deux non-centrale ... 2893
MathProbabilityDensityNoncentralChiSquare ... 2894
MathCumulativeDistributionNoncentralChiSquare ... 2896
MathQuantileNoncentralChiSquare ... 2898
MathRandomNoncentralChiSquare ... 2900
MathMomentsNoncentralChiSquare ... 2901
Distribution exponentielle ... 2902
MathProbabilityDensityExponential ... 2903
MathCumulativeDistributionExponential ... 2905
MathQuantileExponential ... 2907
MathRandomExponential ... 2909
MathMomentsExponential ... 2910
Distribution de Fisher ... 2911
MathProbabilityDensityF ... 2912
MathCumulativeDistributionF ... 2914
MathQuantileF ... 2916
MathRandomF ... 2918
MathMomentsF ... 2919
Distribution de Fisher non-centrale ... 2920
MathProbabilityDensityNoncentralF ... 2921
MathCumulativeDistributionNoncentralF ... 2923
MathQuantileNoncentralF ... 2925
MathRandomNoncentralF ... 2927
MathMomentsNoncentralF ... 2928
Distribution t ... 2929
MathProbabilityDensityT ... 2930
MathCumulativeDistributionT ... 2932
MathQuantileT ... 2934
MathRandomT ... 2936
MathMomentsT ... 2937
Distribution t non-centrale ... 2938
MathProbabilityDensityNoncentralT ... 2939
MathCumulativeDistributionNoncentralT ... 2941
MathQuantileNoncentralT ... 2943
MathRandomNoncentralT ... 2945
MathMomentsNoncentralT ... 2946

© 2000-2024, MetaQuotes Ltd.

Distribution logistique ... 2947
MathProbabilityDensityLogistic ... 2948
MathCumulativeDistributionLogistic ... 2950
MathQuantileLogistic ... 2952
MathRandomLogistic ... 2954
MathMomentsLogistic ... 2955
Distribution de Cauchy ... 2956
MathProbabilityDensityCauchy ... 2957
MathCumulativeDistributionCauchy ... 2959
MathQuantileCauchy ... 2961
MathRandomCauchy ... 2963
MathMomentsCauchy ... 2964
Distribution uniforme ... 2965
MathProbabilityDensityUniform ... 2966
MathCumulativeDistributionUniform ... 2968
MathQuantileUniform ... 2970
MathRandomUniform ... 2972
MathMomentsUniform ... 2973
Distribution Weibull ... 2974
MathProbabilityDensityWeibull ... 2975
MathCumulativeDistributionWeibull ... 2977
MathQuantileWeibull ... 2979
MathRandomWeibull ... 2981
MathMomentsWeibull ... 2982
Distribution binomiale ... 2983
MathProbabilityDensityBinomial ... 2984
MathCumulativeDistributionBinomial ... 2986
MathQuantileBinomial ... 2988
MathRandomBinomial ... 2990
MathMomentsBinomial ... 2991
Distribution binomiale négative ... 2992
MathProbabilityDensityNegativeBinomial ... 2993
MathCumulativeDistributionNegativeBinomial ... 2995
MathQuantileNegativeBinomial ... 2997
MathRandomNegativeBinomial ... 2999
MathMomentsNegativeBinomial ... 3000
Distribution géométrique ... 3001
MathProbabilityDensityGeometric ... 3002
MathCumulativeDistributionGeometric ... 3004
MathQuantileGeometric ... 3006
MathRandomGeometric ... 3008
MathMomentsGeometric ... 3009
Distribution Hypergéometrique ... 3010
MathProbabilityDensityHypergeometric ... 3011
MathCumulativeDistributionHypergeometric ... 3013
MathQuantileHypergeometric ... 3015
MathRandomHypergeometric ... 3017
MathMomentsHypergeometric ... 3018
Distributio de Poisson ... 3019
MathProbabilityDensityPoisson ... 3020
MathCumulativeDistributionPoisson ... 3022
MathQuantilePoisson ... 3024
MathRandomPoisson ... 3026
MathMomentsPoisson ... 3027
Sous-fonctions ... 3028
MathRandomNonZero ... 3033
MathMoments ... 3034
MathPowInt ... 3035

© 2000-2024, MetaQuotes Ltd.

MathFactorial ... 3036
MathTrunc ... 3037
MathRound ... 3038
MathArctan2 ... 3040
MathGamma ... 3042
MathGammaLog ... 3043
MathBeta ... 3044
MathBetaLog ... 3045
MathBetaIncomplete ... 3046
MathGammaIncomplete ... 3047
MathBinomialCoefficient ... 3048
MathBinomialCoefficientLog ... 3049
MathHypergeometric2F2 ... 3050
MathSequence ... 3051
MathSequenceByCount ... 3052
MathReplicate ... 3053
MathReverse ... 3054
MathIdentical ... 3055
MathUnique ... 3056
MathQuickSortAscending ... 3057
MathQuickSortDescending ... 3058
MathQuickSort ... 3059
MathOrder ... 3060
MathBitwiseNot ... 3061
MathBitwiseAnd ... 3062
MathBitwiseOr ... 3063
MathBitwiseXor ... 3064
MathBitwiseShiftL ... 3065
MathBitwiseShiftR ... 3066
MathCumulativeSum ... 3067
MathCumulativeProduct ... 3068
MathCumulativeMin ... 3069
MathCumulativeMax ... 3070
MathSin ... 3071
MathCos ... 3072
MathTan ... 3073
MathArcsin ... 3074
MathArccos ... 3075
MathArctan ... 3076
MathSinPi ... 3077
MathCosPi ... 3078
MathTanPi ... 3079
MathAbs ... 3080
MathCeil ... 3081
MathFloor ... 3082
MathSqrt ... 3083
MathExp ... 3084
MathPow ... 3085
MathLog ... 3086
MathLog2 ... 3087
MathLog10 ... 3088
MathLog1p ... 3089
MathDifference ... 3090
MathSample ... 3092
MathTukeySummary ... 3095
MathRange ... 3096
MathMin ... 3097
MathMax ... 3098

© 2000-2024, MetaQuotes Ltd.

MathSum ... 3099
MathProduct ... 3100
MathStandardDeviation ... 3101
MathAverageDeviation ... 3102
MathMedian ... 3103
MathMean ... 3104
MathVariance ... 3105
MathSkewness ... 3106
MathKurtosis ... 3107
MathExpm1 ... 3108
MathSinh ... 3109
MathCosh ... 3110
MathTanh ... 3111
MathArcsinh ... 3112
MathArccosh ... 3113
MathArctanh ... 3114
MathSignif ... 3115
MathRank ... 3117
MathCorrelationPearson ... 3118
MathCorrelationSpearman ... 3119
MathCorrelationKendall ... 3120
MathQuantile ... 3121
MathProbabilityDensityEmpirical ... 3122
MathCumulativeDistributionEmpirical ... 3123
Logique floue ... 3124
Fonctions d'appartenance ... 3126
CConstantMembershipFunction ... 3128
GetValue ... 3130
CCompositeMembershipFunction ... 3131
CompositionType ... 3133
MembershipFunctions ... 3133
GetValue ... 3133
CDifferencTwoSigmoidalMembershipFunction ... 3135
A1 ... 3137
A2 ... 3137
C1 ... 3138
C2 ... 3138
GetValue ... 3139
CGeneralizedBellShapedMembershipFunction ... 3140
A ... 3142
B ... 3142
C ... 3143
GetValue ... 3143
CNormalCombinationMembershipFunction ... 3144
B1 ... 3146
B2 ... 3146
Sigma1 ... 3147
Sigma2 ... 3147
GetValue ... 3148
CNormalMembershipFunction ... 3149
B ... 3151
Sigma ... 3151
GetValue ... 3152
CP_ShapedMembershipFunction ... 3153
A ... 3155
B ... 3155
C ... 3156
D ... 3156

© 2000-2024, MetaQuotes Ltd.

GetValue ... 3157
CProductTwoSigmoidalMembershipFunctions ... 3158
A1 ... 3160
A2 ... 3160
C1 ... 3161
C2 ... 3161
GetValue ... 3162
CS_ShapedMembershipFunction ... 3163
A ... 3165
B ... 3165
GetValue ... 3166
CSigmoidalMembershipFunction ... 3167
A ... 3169
C ... 3169
GetValue ... 3170
CTrapezoidMembershipFunction ... 3171
X1 ... 3173
X2 ... 3173
X3 ... 3174
X4 ... 3174
GetValue ... 3175
CTriangularMembershipFunction ... 3176
X1 ... 3178
X2 ... 3178
X3 ... 3179
ToNormalMF ... 3179
GetValue ... 3179
CZ_ShapedMembershipFunction ... 3181
A ... 3183
B ... 3183
GetValue ... 3184
IMembershipFunction ... 3185
GetValue ... 3185
Règles de systèmes flous ... 3186
CMamdaniFuzzyRule ... 3187
Conclusion ... 3188
Weight ... 3188
CSugenoFuzzyRule ... 3189
Conclusion ... 3190
CSingleCondition ... 3191
Not ... 3191
Term ... 3192
Var ... 3192
CConditions ... 3194
ConditionsList ... 3194
Not ... 3195
Op ... 3195
CGenericFuzzyRule ... 3196
Conclusion ... 3196
Condition ... 3197
CreateCondition ... 3197
Variables de systèmes flous ... 3199
CFuzzyVariable ... 3200
AddTerm ... 3201
GetTermByName ... 3201
Max ... 3201
Min ... 3202
Terms ... 3202

© 2000-2024, MetaQuotes Ltd.

Values ... 3203
CSugenoVariable ... 3204
Functions ... 3204
GetFuncByName ... 3205
Values ... 3205
Termes flous ... 3206
MembershipFunction ... 3207
Systèmes flous ... 3208
Système de Mamdani ... 3209
AggregationMethod ... 3209
Calculate ... 3210
DefuzzificationMethod ... 3210
EmptyRule ... 3210
ImplicationMethod ... 3210
Output ... 3211
OutputByName ... 3211
ParseRule ... 3211
Rules ... 3211
Système de Sugeno ... 3213
Calculate ... 3213
CreateSugenoFunction ... 3214
EmptyRule ... 3215
Output ... 3215
OutputByName ... 3215
ParseRule ... 3215
Rules ... 3216
OpenCL ... 3217
BufferCreate ... 3219
BufferFree ... 3220
BufferFromArray ... 3221
BufferRead ... 3222
BufferWrite ... 3223
Execute ... 3224
GetContext ... 3225
GetKernel ... 3226
GetKernelName ... 3227
GetProgram ... 3228
Initialize ... 3229
KernelCreate ... 3230
KernelFree ... 3231
SetArgument ... 3232
SetArgumentBuffer ... 3233
SetArgumentLocalMemory ... 3234
SetBuffersCount ... 3235
SetKernelsCount ... 3236
Shutdown ... 3237
SupportDouble ... 3238
Classe de Base CObject ... 3239
Prev ... 3240
Prev ... 3241
Next ... 3242
Next ... 3243
Compare ... 3244
Save ... 3246
Load ... 3248
Type ... 3250
Collections de Données ... 3251
CArray ... 3253

© 2000-2024, MetaQuotes Ltd.

Step ... 3255
Step ... 3256
Total ... 3257
Available ... 3258
Max ... 3259
IsSorted ... 3260
SortMode ... 3261
Clear ... 3262
Sort ... 3263
Save ... 3264
Load ... 3265
CArrayChar ... 3266
Reserve ... 3269
Resize ... 3270
Shutdown ... 3271
Add ... 3272
AddArray ... 3273
AddArray ... 3274
Insert ... 3276
InsertArray ... 3277
InsertArray ... 3278
AssignArray ... 3280
AssignArray ... 3281
Update ... 3283
Shift ... 3284
Delete ... 3285
DeleteRange ... 3286
At ... 3287
CompareArray ... 3289
CompareArray ... 3290
InsertSort ... 3291
Search ... 3292
SearchGreat ... 3293
SearchLess ... 3294
SearchGreatOrEqual ... 3295
SearchLessOrEqual ... 3296
SearchFirst ... 3297
SearchLast ... 3298
SearchLinear ... 3299
Save ... 3300
Load ... 3301
Type ... 3303
CArrayShort ... 3304
Reserve ... 3307
Resize ... 3308
Shutdown ... 3309
Add ... 3310
AddArray ... 3311
AddArray ... 3312
Insert ... 3314
InsertArray ... 3315
InsertArray ... 3316
AssignArray ... 3318
AssignArray ... 3319
Update ... 3321
Shift ... 3322
Delete ... 3323
DeleteRange ... 3324

© 2000-2024, MetaQuotes Ltd.

At ... 3325
CompareArray ... 3327
CompareArray ... 3328
InsertSort ... 3329
Search ... 3330
SearchGreat ... 3331
SearchLess ... 3332
SearchGreatOrEqual ... 3333
SearchLessOrEqual ... 3334
SearchFirst ... 3335
SearchLast ... 3336
SearchLinear ... 3337
Save ... 3338
Load ... 3340
Type ... 3342
CArrayInt ... 3343
Reserve ... 3346
Resize ... 3347
Shutdown ... 3348
Add ... 3349
AddArray ... 3350
AddArray ... 3351
Insert ... 3353
InsertArray ... 3354
InsertArray ... 3355
AssignArray ... 3357
AssignArray ... 3358
Update ... 3360
Shift ... 3361
Delete ... 3362
DeleteRange ... 3363
At ... 3364
CompareArray ... 3366
CompareArray ... 3367
InsertSort ... 3368
Search ... 3369
SearchGreat ... 3370
SearchLess ... 3371
SearchGreatOrEqual ... 3372
SearchLessOrEqual ... 3373
SearchFirst ... 3374
SearchLast ... 3375
SearchLinear ... 3376
Save ... 3377
Load ... 3379
Type ... 3381
CArrayLong ... 3382
Reserve ... 3385
Resize ... 3386
Shutdown ... 3387
Add ... 3388
AddArray ... 3389
AddArray ... 3390
Insert ... 3392
InsertArray ... 3393
InsertArray ... 3394
AssignArray ... 3396
AssignArray ... 3397

© 2000-2024, MetaQuotes Ltd.

Update ... 3399
Shift ... 3400
Delete ... 3401
DeleteRange ... 3402
At ... 3403
CompareArray ... 3405
CompareArray ... 3406
InsertSort ... 3407
Search ... 3408
SearchGreat ... 3409
SearchLess ... 3410
SearchGreatOrEqual ... 3411
SearchLessOrEqual ... 3412
SearchFirst ... 3413
SearchLast ... 3414
SearchLinear ... 3415
Save ... 3416
Load ... 3418
Type ... 3420
CArrayFloat ... 3421
Delta ... 3424
Reserve ... 3425
Resize ... 3426
Shutdown ... 3427
Add ... 3428
AddArray ... 3429
AddArray ... 3430
Insert ... 3432
InsertArray ... 3433
InsertArray ... 3434
AssignArray ... 3436
AssignArray ... 3437
Update ... 3439
Shift ... 3440
Delete ... 3441
DeleteRange ... 3442
At ... 3443
CompareArray ... 3445
CompareArray ... 3446
InsertSort ... 3447
Search ... 3448
SearchGreat ... 3449
SearchLess ... 3450
SearchGreatOrEqual ... 3451
SearchLessOrEqual ... 3452
SearchFirst ... 3453
SearchLast ... 3454
SearchLinear ... 3455
Save ... 3456
Load ... 3458
Type ... 3460
CArrayDouble ... 3461
Delta ... 3464
Reserve ... 3465
Resize ... 3466
Shutdown ... 3467
Add ... 3468
AddArray ... 3469

© 2000-2024, MetaQuotes Ltd.

AddArray ... 3470
Insert ... 3472
InsertArray ... 3473
InsertArray ... 3474
AssignArray ... 3476
AssignArray ... 3477
Update ... 3479
Shift ... 3480
Delete ... 3481
DeleteRange ... 3482
At ... 3483
CompareArray ... 3485
CompareArray ... 3486
Minimum ... 3487
Maximum ... 3488
InsertSort ... 3489
Search ... 3490
SearchGreat ... 3491
SearchLess ... 3492
SearchGreatOrEqual ... 3493
SearchLessOrEqual ... 3494
SearchFirst ... 3495
SearchLast ... 3496
SearchLinear ... 3497
Save ... 3498
Load ... 3500
Type ... 3502
CArrayString ... 3503
Reserve ... 3506
Resize ... 3507
Shutdown ... 3508
Add ... 3509
AddArray ... 3510
AddArray ... 3511
Insert ... 3513
InsertArray ... 3514
InsertArray ... 3515
AssignArray ... 3517
AssignArray ... 3518
Update ... 3520
Shift ... 3521
Delete ... 3522
DeleteRange ... 3523
At ... 3524
CompareArray ... 3526
CompareArray ... 3527
InsertSort ... 3528
Search ... 3529
SearchGreat ... 3530
SearchLess ... 3531
SearchGreatOrEqual ... 3532
SearchLessOrEqual ... 3533
SearchFirst ... 3534
SearchLast ... 3535
SearchLinear ... 3536
Save ... 3537
Load ... 3539
Type ... 3541

© 2000-2024, MetaQuotes Ltd.

CArrayObj ... 3542
FreeMode ... 3547
FreeMode ... 3548
Reserve ... 3550
Resize ... 3551
Clear ... 3552
Shutdown ... 3553
CreateElement ... 3554
Add ... 3556
AddArray ... 3557
Insert ... 3560
InsertArray ... 3562
AssignArray ... 3564
Update ... 3566
Shift ... 3567
Detach ... 3568
Delete ... 3569
DeleteRange ... 3570
At ... 3571
CompareArray ... 3572
InsertSort ... 3573
Search ... 3574
SearchGreat ... 3575
SearchLess ... 3576
SearchGreatOrEqual ... 3577
SearchLessOrEqual ... 3578
SearchFirst ... 3579
SearchLast ... 3580
Save ... 3581
Load ... 3582
Type ... 3584
CList ... 3585
FreeMode ... 3588
FreeMode ... 3589
Total ... 3591
IsSorted ... 3592
SortMode ... 3593
CreateElement ... 3594
Add ... 3595
Insert ... 3596
DetachCurrent ... 3598
DeleteCurrent ... 3599
Delete ... 3600
Clear ... 3601
IndexOf ... 3602
GetNodeAtIndex ... 3603
GetFirstNode ... 3604
GetPrevNode ... 3605
GetCurrentNode ... 3606
GetNextNode ... 3607
GetLastNode ... 3608
Sort ... 3609
MoveToIndex ... 3610
Exchange ... 3611
CompareList ... 3612
Search ... 3613
Save ... 3614
Load ... 3616

© 2000-2024, MetaQuotes Ltd.

Type ... 3618
CTreeNode ... 3619
Owner ... 3624
Left ... 3625
Right ... 3626
Balance ... 3627
BalanceL ... 3628
BalanceR ... 3629
CreateSample ... 3630
RefreshBalance ... 3631
GetNext ... 3632
SaveNode ... 3633
LoadNode ... 3634
Type ... 3635
CTree ... 3636
Root ... 3642
CreateElement ... 3643
Insert ... 3644
Detach ... 3645
Delete ... 3646
Clear ... 3647
Find ... 3648
Save ... 3649
Load ... 3650
Type ... 3651
Collections de Données Génériques ... 3652
ICollection<T> ... 3655
Add ... 3656
Count ... 3657
Contains ... 3658
CopyTo ... 3659
Clear ... 3660
Remove ... 3661
IEqualityComparable<T> ... 3662
Equals ... 3663
HashCode ... 3664
IComparable<T> ... 3665
Compare ... 3666
IComparer<T> ... 3667
Compare ... 3668
IEqualityComparer<T> ... 3669
Equals ... 3670
HashCode ... 3671
IList<T> ... 3672
TryGetValue ... 3673
TrySetValue ... 3674
Insert ... 3675
IndexOf ... 3676
LastIndexOf ... 3677
RemoveAt ... 3678
IMap<TKey,TValue> ... 3679
Add ... 3680
Contains ... 3681
Remove ... 3682
TryGetValue ... 3683
TrySetValue ... 3684
CopyTo ... 3685
ISet<T> ... 3686

© 2000-2024, MetaQuotes Ltd.

ExceptWith ... 3688
IntersectWith ... 3689
SymmetricExceptWith ... 3690
UnionWith ... 3691
IsProperSubsetOf ... 3692
IsProperSupersetOf ... 3693
IsSubsetOf ... 3694
IsSupersetOf ... 3695
Overlaps ... 3696
SetEquals ... 3697
CDefaultComparer<T> ... 3698
Compare ... 3699
CDefaultEqualityComparer<T> ... 3700
Equals ... 3701
HashCode ... 3702
CRedBlackTreeNode<T> ... 3703
Value ... 3704
Parent ... 3705
Left ... 3706
Right ... 3707
Color ... 3708
IsLeaf ... 3709
CreateEmptyNode ... 3710
CLinkedListNode<T> ... 3711
List ... 3712
Next ... 3713
Previous ... 3714
Value ... 3715
CKeyValuePair<TKey,TValue> ... 3716
Key ... 3717
Value ... 3718
Clone ... 3719
Compare ... 3720
Equals ... 3721
HashCode ... 3722
CArrayList<T> ... 3723
Capacity ... 3725
Count ... 3726
Contains ... 3727
TrimExcess ... 3728
TryGetValue ... 3729
TrySetValue ... 3730
Add ... 3731
AddRange ... 3732
Insert ... 3733
InsertRange ... 3734
CopyTo ... 3735
BinarySearch ... 3736
IndexOf ... 3737
LastIndexOf ... 3738
Clear ... 3739
Remove ... 3740
RemoveAt ... 3741
RemoveRange ... 3742
Reverse ... 3743
Sort ... 3744
CHashMap<TKey,TValue> ... 3745
Add ... 3747

© 2000-2024, MetaQuotes Ltd.

Count ... 3748
Comparer ... 3749
Contains ... 3750
ContainsKey ... 3751
ContainsValue ... 3752
CopyTo ... 3753
Clear ... 3754
Remove ... 3755
TryGetValue ... 3756
TrySetValue ... 3757
CHashSet<T> ... 3758
Add ... 3760
Count ... 3761
Contains ... 3762
Comparer ... 3763
TrimExcess ... 3764
CopyTo ... 3765
Clear ... 3766
Remove ... 3767
ExceptWith ... 3768
IntersectWith ... 3769
SymmetricExceptWith ... 3770
UnionWith ... 3771
IsProperSubsetOf ... 3772
IsProperSupersetOf ... 3773
IsSubsetOf ... 3774
IsSupersetOf ... 3775
Overlaps ... 3776
SetEquals ... 3777
CLinkedList<T> ... 3778
Add ... 3780
AddAfter ... 3781
AddBefore ... 3782
AddFirst ... 3783
AddLast ... 3784
Count ... 3785
Head ... 3786
First ... 3787
Last ... 3788
Contains ... 3789
CopyTo ... 3790
Clear ... 3791
Remove ... 3792
RemoveFirst ... 3793
RemoveLast ... 3794
Find ... 3795
FindLast ... 3796
CQueue<T> ... 3797
Add ... 3798
Enqueue ... 3799
Count ... 3800
Contains ... 3801
TrimExcess ... 3802
CopyTo ... 3803
Clear ... 3804
Remove ... 3805
Dequeue ... 3806
Peek ... 3807

© 2000-2024, MetaQuotes Ltd.

CRedBlackTree<T> ... 3808
Add ... 3810
Count ... 3811
Root ... 3812
Contains ... 3813
Comparer ... 3814
TryGetMin ... 3815
TryGetMax ... 3816
CopyTo ... 3817
Clear ... 3818
Remove ... 3819
RemoveMin ... 3820
RemoveMax ... 3821
Find ... 3822
FindMin ... 3823
FindMax ... 3824
CSortedMap<TKey, TValue> ... 3825
Add ... 3827
Count ... 3828
Comparer ... 3829
Contains ... 3830
ContainsKey ... 3831
ContainsValue ... 3832
CopyTo ... 3833
Clear ... 3834
Remove ... 3835
TryGetValue ... 3836
TrySetValue ... 3837
CSortedSet<T> ... 3838
Add ... 3840
Count ... 3841
Contains ... 3842
Comparer ... 3843
TryGetMin ... 3844
TryGetMax ... 3845
CopyTo ... 3846
Clear ... 3847
Remove ... 3848
ExceptWith ... 3849
IntersectWith ... 3850
SymmetricExceptWith ... 3851
UnionWith ... 3852
IsProperSubsetOf ... 3853
IsProperSupersetOf ... 3854
IsSubsetOf ... 3855
IsSupersetOf ... 3856
Overlaps ... 3857
SetEquals ... 3858
GetViewBetween ... 3859
GetReverse ... 3860
CStack<T> ... 3861
Add ... 3862
Count ... 3863
Contains ... 3864
TrimExcess ... 3865
CopyTo ... 3866
Clear ... 3867
Remove ... 3868

© 2000-2024, MetaQuotes Ltd.

Push ... 3869
Peek ... 3870
Pop ... 3871
ArrayBinarySearch<T> ... 3872
ArrayIndexOf<T> ... 3873
ArrayLastIndexOf<T> ... 3874
ArrayReverse<T> ... 3875
Compare ... 3876
Equals<T> ... 3879
GetHashCode ... 3880
Fichiers ... 3883
CFile ... 3884
Handle ... 3886
Filename ... 3887
Flags ... 3888
SetUnicode ... 3889
SetCommon ... 3890
Open ... 3891
Close ... 3892
Delete ... 3893
IsExist ... 3894
Copy ... 3895
Move ... 3896
Size ... 3897
Tell ... 3898
Seek ... 3899
Flush ... 3900
IsEnding ... 3901
IsLineEnding ... 3902
FolderCreate ... 3903
FolderDelete ... 3904
FolderClean ... 3905
FileFindFirst ... 3906
FileFindNext ... 3907
FileFindClose ... 3908
CFileBin ... 3909
Open ... 3911
WriteChar ... 3912
WriteShort ... 3913
WriteInteger ... 3914
WriteLong ... 3915
WriteFloat ... 3916
WriteDouble ... 3917
WriteString ... 3918
WriteCharArray ... 3919
WriteShortArray ... 3920
WriteIntegerArray ... 3921
WriteLongArray ... 3922
WriteFloatArray ... 3923
WriteDoubleArray ... 3924
WriteObject ... 3925
ReadChar ... 3926
ReadShort ... 3927
ReadInteger ... 3928
ReadLong ... 3929
ReadFloat ... 3930
ReadDouble ... 3931
ReadString ... 3932

© 2000-2024, MetaQuotes Ltd.

ReadCharArray ... 3933
ReadShortArray ... 3934
ReadIntegerArray ... 3935
ReadLongArray ... 3936
ReadFloatArray ... 3937
ReadDoubleArray ... 3938
ReadObject ... 3939
CFileTxt ... 3940
Open ... 3941
WriteString ... 3942
ReadString ... 3943
Chaînes de Caractères ... 3944
CString ... 3945
Str ... 3947
Len ... 3948
Copy ... 3949
Fill ... 3950
Assign ... 3951
Append ... 3952
Insert ... 3953
Compare ... 3954
CompareNoCase ... 3955
Left ... 3956
Right ... 3957
Mid ... 3958
Trim ... 3959
TrimLeft ... 3960
TrimRight ... 3961
Clear ... 3962
ToUpper ... 3963
ToLower ... 3964
Reverse ... 3965
Find ... 3966
FindRev ... 3967
Remove ... 3968
Replace ... 3969
Objets Graphiques ... 3970
CChartObject ... 3971
ChartId ... 3974
Window ... 3975
Name ... 3976
NumPoints ... 3977
Attach ... 3978
SetPoint ... 3979
Delete ... 3980
Detach ... 3981
ShiftObject ... 3982
ShiftPoint ... 3983
Time ... 3984
Price ... 3986
Color ... 3988
Style ... 3989
Width ... 3990
Background ... 3991
Selected ... 3992
Selectable ... 3993
Description ... 3994
Tooltip ... 3995

© 2000-2024, MetaQuotes Ltd.

Timeframes ... 3996
Z_Order ... 3997
CreateTime ... 3998
LevelsCount ... 3999
LevelColor ... 4000
LevelStyle ... 4002
LevelWidth ... 4004
LevelValue ... 4006
LevelDescription ... 4008
GetInteger ... 4010
SetInteger ... 4012
GetDouble ... 4014
SetDouble ... 4016
GetString ... 4018
SetString ... 4020
Save ... 4022
Load ... 4023
Type ... 4024
Objets Lignes ... 4025
CChartObjectVLine ... 4026
Create ... 4027
Type ... 4028
CChartObjectHLine ... 4029
Create ... 4030
Type ... 4031
CChartObjectTrend ... 4032
Create ... 4034
RayLeft ... 4035
RayRight ... 4036
Save ... 4037
Load ... 4038
Type ... 4039
CChartObjectTrendByAngle ... 4040
Create ... 4042
Angle ... 4043
Type ... 4044
CChartObjectCycles ... 4045
Create ... 4046
Type ... 4047
Objets Canaux ... 4048
CChartObjectChannel ... 4049
Create ... 4051
Type ... 4052
CChartObjectRegression ... 4053
Create ... 4055
Type ... 4056
CChartObjectStdDevChannel ... 4057
Create ... 4059
Deviation ... 4060
Save ... 4061
Load ... 4062
Type ... 4063
CChartObjectPitchfork ... 4064
Create ... 4066
Type ... 4067
Outils de Gann ... 4068
CChartObjectGannLine ... 4069
Create ... 4071

© 2000-2024, MetaQuotes Ltd.

PipsPerBar ... 4072
Save ... 4073
Load ... 4074
Type ... 4075
CChartObjectGannFan ... 4076
Create ... 4078
PipsPerBar ... 4079
Downtrend ... 4080
Save ... 4081
Load ... 4082
Type ... 4083
CChartObjectGannGrid ... 4084
Create ... 4086
PipsPerBar ... 4087
Downtrend ... 4088
Save ... 4089
Load ... 4090
Type ... 4091
Outils de Fibonacci ... 4092
CChartObjectFibo ... 4093
Create ... 4095
Type ... 4096
CChartObjectFiboTimes ... 4097
Create ... 4098
Type ... 4099
CChartObjectFiboFan ... 4100
Create ... 4101
Type ... 4102
CChartObjectFiboArc ... 4103
Create ... 4105
Scale ... 4106
Ellipse ... 4107
Save ... 4108
Load ... 4109
Type ... 4110
CChartObjectFiboChannel ... 4111
Create ... 4113
Type ... 4114
CChartObjectFiboExpansion ... 4115
Create ... 4117
Type ... 4118
Outils d'Elliott ... 4119
CChartObjectElliottWave3 ... 4120
Create ... 4122
Degree ... 4123
Lignes ... 4124
Save ... 4125
Load ... 4126
Type ... 4127
CChartObjectElliottWave5 ... 4128
Create ... 4130
Type ... 4132
Objets Formes ... 4133
CChartObjectRectangle ... 4134
Create ... 4135
Type ... 4136
CChartObjectTriangle ... 4137
Create ... 4138

© 2000-2024, MetaQuotes Ltd.

Type ... 4139
CChartObjectEllipse ... 4140
Create ... 4141
Type ... 4142
Objets Flèches ... 4143
CChartObjectArrow ... 4144
Create ... 4146
ArrowCode ... 4148
Anchor ... 4150
Save ... 4152
Load ... 4153
Type ... 4154
Flèches prédéfinies ... 4155
Create ... 4157
ArrowCode ... 4159
Type ... 4160
Objets Contrôles ... 4161
CChartObjectText ... 4162
Create ... 4164
Angle ... 4165
Font ... 4166
FontSize ... 4167
Anchor ... 4168
Save ... 4169
Load ... 4170
Type ... 4171
CChartObjectLabel ... 4172
Create ... 4174
X_Distance ... 4175
Y_Distance ... 4176
X_Size ... 4177
Y_Size ... 4178
Corner ... 4179
Time ... 4180
Price ... 4181
Save ... 4182
Load ... 4183
Type ... 4184
CChartObjectEdit ... 4185
Create ... 4187
TextAlign ... 4188
X_Size ... 4189
Y_Size ... 4190
BackColor ... 4191
BorderColor ... 4192
ReadOnly ... 4193
Angle ... 4194
Save ... 4195
Load ... 4196
Type ... 4197
CChartObjectButton ... 4198
Etat ... 4200
Save ... 4201
Load ... 4202
Type ... 4203
CChartObjectSubChart ... 4204
Create ... 4206
X_Distance ... 4207

© 2000-2024, MetaQuotes Ltd.

Y_Distance ... 4208
Corner ... 4209
X_Size ... 4210
Y_Size ... 4211
Symbol ... 4212
Period ... 4213
Scale ... 4214
DateScale ... 4215
PriceScale ... 4216
Time ... 4217
Price ... 4218
Save ... 4219
Load ... 4220
Type ... 4221
CChartObjectBitmap ... 4222
Create ... 4224
BmpFile ... 4225
X_Offset ... 4226
Y_Offset ... 4227
Save ... 4228
Load ... 4229
Type ... 4230
CChartObjectBmpLabel ... 4231
Create ... 4233
X_Distance ... 4234
Y_Distance ... 4235
X_Offset ... 4236
Y_Offset ... 4237
Corner ... 4238
X_Size ... 4239
Y_Size ... 4240
BmpFileOn ... 4241
BmpFileOff ... 4242
Etat ... 4243
Time ... 4244
Price ... 4245
Save ... 4246
Load ... 4247
Type ... 4248
CChartObjectRectLabel ... 4249
Create ... 4251
X_Size ... 4252
Y_Size ... 4253
BackColor ... 4254
Angle ... 4255
BorderType ... 4256
Save ... 4257
Load ... 4258
Type ... 4259
Graphiques personnalisés ... 4260
CCanvas ... 4261
Attach ... 4265
Arc ... 4266
Pie ... 4270
FillPolygon ... 4274
FillEllipse ... 4275
GetDefaultColor ... 4276
ChartObjectName ... 4277

© 2000-2024, MetaQuotes Ltd.

Circle 4278
CircleAA 4279
CircleWu 4280
Create 4281
CreateBitmap 4282
CreateBitmapLabel 4284
Destroy 4286
Ellipse 4287
EllipseAA 4288
EllipseWu 4289
Erase 4290
Fill 4291
FillCircle 4292
FillRectangle 4293
FillTriangle 4294
FontAngleGet 4295
FontAngleSet 4296
FontFlagsGet 4297
FontFlagsSet 4298
FontGet 4299
FontNameGet 4300
FontNameSet 4301
FontSet 4302
FontSizeGet 4303
FontSizeSet 4304
Height 4305
Line 4306
LineAA 4307
LineWu 4308
LineHorizontal 4309
LineVertical 4310
LineStyleSet 4311
LineThick 4312
LineThickVertical 4314
LineThickHorizontal 4315
LoadFromFile 4316
PixelGet 4317
PixelSet 4318
PixelSetAA 4319
Polygon 4320
PolygonAA 4321
PolygonWu 4322
PolygonThick 4323
PolygonSmooth 4324
Polyline 4325
PolylineSmooth 4326
PolylineThick 4327
PolylineWu 4328
PolylineAA 4329
Rectangle 4330
Resize 4331
ResourceName 4332
TextHeight 4333
TextOut 4334
TextSize 4335
TextWidth 4336
TransparentLevelSet 4337
Triangle 4338

© 2000-2024, MetaQuotes Ltd.

TriangleAA ... 4339
TriangleWu ... 4340
Update ... 4341
Width ... 4342
CChartCanvas ... 4343
ColorBackground ... 4347
ColorBorder ... 4348
ColorText ... 4349
ColorGrid ... 4350
MaxData ... 4351
MaxDescrLen ... 4352
ShowFlags ... 4353
IsShowLegend ... 4354
IsShowScaleLeft ... 4355
IsShowScaleRight ... 4356
IsShowScaleTop ... 4357
IsShowScaleBottom ... 4358
IsShowGrid ... 4359
IsShowDescriptors ... 4360
IsShowPercent ... 4361
VScaleMin ... 4362
VScaleMax ... 4363
NumGrid ... 4364
DataOffset ... 4365
DataTotal ... 4366
DrawDescriptors ... 4367
DrawData ... 4368
Create ... 4369
AllowedShowFlags ... 4370
ShowLegend ... 4371
ShowScaleLeft ... 4372
ShowScaleRight ... 4373
ShowScaleTop ... 4374
ShowScaleBottom ... 4375
ShowGrid ... 4376
ShowDescriptors ... 4377
ShowValue ... 4378
ShowPercent ... 4379
LegendAlignment ... 4380
Accumulative ... 4381
VScaleParams ... 4382
DescriptorUpdate ... 4383
ColorUpdate ... 4384
ValuesCheck ... 4385
Redraw ... 4386
DrawBackground ... 4387
DrawLegend ... 4388
DrawLegendVertical ... 4389
DrawLegendHorizontal ... 4390
CalcScales ... 4391
DrawScales ... 4392
DrawScaleLeft ... 4393
DrawScaleRight ... 4394
DrawScaleTop ... 4395
DrawScaleBottom ... 4396
DrawGrid ... 4397
DrawChart ... 4398
CHistogramChart ... 4399

© 2000-2024, MetaQuotes Ltd.

Gradient ... 4404
BarGap ... 4405
BarMinSize ... 4406
BarBorder ... 4407
Create ... 4408
SeriesAdd ... 4409
SeriesInsert ... 4410
SeriesUpdate ... 4411
SeriesDelete ... 4412
ValueUpdate ... 4413
DrawData ... 4414
DrawBar ... 4415
GradientBrush ... 4416
CLineChart ... 4417
Filled ... 4421
Create ... 4422
SeriesAdd ... 4423
SeriesInsert ... 4424
SeriesUpdate ... 4425
SeriesDelete ... 4426
ValueUpdate ... 4427
DrawChart ... 4428
DrawData ... 4429
CalcArea ... 4430
CPieChart ... 4431
Create ... 4435
SeriesSet ... 4436
ValueAdd ... 4437
ValueInsert ... 4438
ValueUpdate ... 4439
ValueDelete ... 4440
DrawChart ... 4441
DrawPie ... 4442
LabelMake ... 4443
Graphiques 3D ... 4444
CCanvas3D ... 4445
AmbientColorGet ... 4447
AmbientColorSet ... 4448
Attach ... 4449
Create ... 4450
Destroy ... 4451
DXContext ... 4452
DXDispatcher ... 4453
InputScene ... 4454
LightColorGet ... 4455
LightColorSet ... 4456
LightDirectionGet ... 4457
LightDirectionSet ... 4458
ObjectAdd ... 4459
ProjectionMatrixGet ... 4460
ProjectionMatrixSet ... 4461
Render ... 4462
RenderBegin ... 4463
RenderEnd ... 4464
ViewMatrixGet ... 4465
ViewMatrixSet ... 4466
ViewPositionSet ... 4467
ViewRotationSet ... 4468

© 2000-2024, MetaQuotes Ltd.

ViewTargetSet ... 4469
ViewUpDirectionSet ... 4470
Graphiques des prix ... 4471
ChartID ... 4476
Mode ... 4477
Foreground ... 4478
Shift ... 4479
ShiftSize ... 4480
AutoScroll ... 4481
Scale ... 4482
ScaleFix ... 4483
ScaleFix_11 ... 4484
FixedMax ... 4485
FixedMin ... 4486
PointsPerBar ... 4487
ScalePPB ... 4488
ShowOHLC ... 4489
ShowLineBid ... 4490
ShowLineAsk ... 4491
ShowLastLine ... 4492
ShowPeriodSep ... 4493
ShowGrid ... 4494
ShowVolumes ... 4495
ShowObjectDescr ... 4496
ShowDateScale ... 4497
ShowPriceScale ... 4498
ColorBackground ... 4499
ColorForeground ... 4500
ColorGrid ... 4501
ColorBarUp ... 4502
ColorBarDown ... 4503
ColorCandleBull ... 4504
ColorCandleBear ... 4505
ColorChartLine ... 4506
ColorVolumes ... 4507
ColorLineBid ... 4508
ColorLineAsk ... 4509
ColorLineLast ... 4510
ColorStopLevels ... 4511
VisibleBars ... 4512
WindowsTotal ... 4513
WindowIsVisible ... 4514
WindowHandle ... 4515
FirstVisibleBar ... 4516
WidthInBars ... 4517
WidthInPixels ... 4518
HeightInPixels ... 4519
PriceMin ... 4520
PriceMax ... 4521
Attach ... 4522
FirstChart ... 4523
NextChart ... 4524
Open ... 4525
Detach ... 4526
Close ... 4527
BringToTop ... 4528
EventObjectCreate ... 4529
EventObjectDelete ... 4530

© 2000-2024, MetaQuotes Ltd.

IndicatorAdd ... 4531
IndicatorDelete ... 4532
IndicatorsTotal ... 4533
IndicatorName ... 4534
Navigate ... 4535
Symbol ... 4536
Period ... 4537
Redraw ... 4538
GetInteger ... 4539
SetInteger ... 4540
GetDouble ... 4541
SetDouble ... 4542
GetString ... 4543
SetString ... 4544
SetSymbolPeriod ... 4545
ApplyTemplate ... 4546
ScreenShot ... 4547
WindowOnDropped ... 4548
PriceOnDropped ... 4549
TimeOnDropped ... 4550
XOnDropped ... 4551
YOnDropped ... 4552
Save ... 4553
Load ... 4554
Type ... 4555
Graphiques Scientifiques ... 4556
GraphPlot ... 4557
CAxis ... 4561
AutoScale ... 4563
Min ... 4564
Max ... 4565
Step ... 4566
Name ... 4567
Color ... 4568
ValuesSize ... 4569
ValuesWidth ... 4570
ValuesFormat ... 4571
ValuesDateTimeMode ... 4572
ValuesFunctionFormat ... 4573
ValuesFunctionFormatCBData ... 4575
NameSize ... 4576
ZeroLever ... 4577
DefaultStep ... 4578
MaxLabels ... 4579
MinGrace ... 4580
MaxGrace ... 4581
SelectAxisScale ... 4582
CColorGenerator ... 4583
Next ... 4584
Reset ... 4585
CCurve ... 4586
Type ... 4588
Name ... 4589
Color ... 4590
XMax ... 4591
XMin ... 4592
YMax ... 4593
YMin ... 4594

© 2000-2024, MetaQuotes Ltd.

Size ... 4595
PointsSize ... 4596
PointsFill ... 4597
PointsColor ... 4598
GetX ... 4599
GetY ... 4600
LinesStyle ... 4601
LinesIsSmooth ... 4602
LinesSmoothTension ... 4603
LinesSmoothStep ... 4604
LinesWidth ... 4605
LinesEndStyle ... 4607
HistogramWidth ... 4608
CustomPlotCBData ... 4609
CustomPlotFunction ... 4610
PointsType ... 4614
StepsDimension ... 4615
TrendLineCoefficients ... 4616
TrendLineColor ... 4617
TrendLineVisible ... 4618
Update ... 4620
Visible ... 4622
CGraphic ... 4623
Create ... 4626
Destroy ... 4627
Update ... 4628
ChartObjectName ... 4629
ResourceName ... 4630
XAxis ... 4631
YAxis ... 4632
GapSize ... 4633
BackgroundColor ... 4634
BackgroundMain ... 4635
BackgroundMainSize ... 4636
BackgroundMainColor ... 4637
BackgroundSub ... 4638
BackgroundSubSize ... 4639
BackgroundSubColor ... 4640
GridLineColor ... 4641
GridBackgroundColor ... 4642
GridCircleRadius ... 4643
GridCircleColor ... 4644
GridHasCircle ... 4645
GridAxisLineColor ... 4646
HistoryNameWidth ... 4647
HistoryNameSize ... 4648
HistorySymbolSize ... 4649
TextAdd ... 4650
LineAdd ... 4651
CurveAdd ... 4652
CurvePlot ... 4655
CurvePlotAll ... 4656
CurveGetByIndex ... 4657
CurveGetByName ... 4658
CurvesTotal ... 4659
CurveRemoveByName ... 4660
CurveRemoveByIndex ... 4661
MarksToAxisAdd ... 4662

© 2000-2024, MetaQuotes Ltd.

MajorMarkSize ... 4663
FontSet ... 4664
FontGet ... 4665
Attach ... 4666
CalculateMaxMinValues ... 4667
Height ... 4668
IndentDown ... 4669
IndentLeft ... 4670
IndentRight ... 4671
IndentUp ... 4672
Redraw ... 4673
ResetParameters ... 4674
ScaleX ... 4675
ScaleY ... 4676
SetDefaultParameters ... 4677
Width ... 4678
Indicateurs ... 4679
Classes de base ... 4680
CSpreadBuffer ... 4681
Size ... 4683
SetSymbolPeriod ... 4684
At ... 4685
Refresh ... 4686
RefreshCurrent ... 4687
CTimeBuffer ... 4688
Size ... 4690
SetSymbolPeriod ... 4691
At ... 4692
Refresh ... 4693
RefreshCurrent ... 4694
CTickVolumeBuffer ... 4695
Size ... 4697
SetSymbolPeriod ... 4698
At ... 4699
Refresh ... 4700
RefreshCurrent ... 4701
CRealVolumeBuffer ... 4702
Size ... 4704
SetSymbolPeriod ... 4705
At ... 4706
Refresh ... 4707
RefreshCurrent ... 4708
CDoubleBuffer ... 4709
Size ... 4711
SetSymbolPeriod ... 4712
At ... 4713
Refresh ... 4714
RefreshCurrent ... 4715
COpenBuffer ... 4716
Refresh ... 4717
RefreshCurrent ... 4718
CHighBuffer ... 4719
Refresh ... 4720
RefreshCurrent ... 4721
CLowBuffer ... 4722
Refresh ... 4723
RefreshCurrent ... 4724
CCloseBuffer ... 4725

© 2000-2024, MetaQuotes Ltd.

Refresh ... 4726
RefreshCurrent ... 4727
CIndicatorBuffer ... 4728
Offset ... 4730
Name ... 4731
At ... 4732
Refresh ... 4733
RefreshCurrent ... 4734
CSeries ... 4735
Name ... 4737
BuffersTotal ... 4738
Timeframe ... 4739
Symbol ... 4740
Period ... 4741
RefreshCurrent ... 4742
BufferSize ... 4743
BufferResize ... 4744
Refresh ... 4745
PeriodDescription ... 4746
CPriceSeries ... 4747
BufferResize ... 4749
GetData ... 4750
Refresh ... 4751
MinIndex ... 4752
MinValue ... 4753
MaxIndex ... 4754
MaxValue ... 4755
CIndicator ... 4756
Handle ... 4760
Status ... 4761
FullRelease ... 4762
Create ... 4763
BufferResize ... 4764
BarsCalculated ... 4765
GetData ... 4766
Refresh ... 4769
Minimum ... 4770
MinValue ... 4771
Maximum ... 4772
MaxValue ... 4773
MethodDescription ... 4774
PriceDescription ... 4775
VolumeDescription ... 4776
AddToChart ... 4777
DeleteFromChart ... 4778
CIndicators ... 4779
Create ... 4780
Refresh ... 4781
Classes des séries de données ... 4782
CiSpread ... 4783
Create ... 4785
BufferResize ... 4786
GetData ... 4787
Refresh ... 4789
CiTime ... 4790
Create ... 4792
BufferResize ... 4793
GetData ... 4794

© 2000-2024, MetaQuotes Ltd.

Refresh ... 4796
CiTickVolume ... 4797
Create ... 4799
BufferResize ... 4800
GetData ... 4801
Refresh ... 4803
CiRealVolume ... 4804
Create ... 4806
BufferResize ... 4807
GetData ... 4808
Refresh ... 4810
CiOpen ... 4811
Create ... 4813
GetData ... 4814
CiHigh ... 4816
Create ... 4818
GetData ... 4819
CiLow ... 4821
Create ... 4823
GetData ... 4824
CiClose ... 4826
Create ... 4828
GetData ... 4829
Indicateurs de Tendance ... 4831
CiADX ... 4832
MaPeriod ... 4834
Create ... 4835
Main ... 4836
Plus ... 4837
Minus ... 4838
Type ... 4839
CiADXWilder ... 4840
MaPeriod ... 4842
Create ... 4843
Main ... 4844
Plus ... 4845
Minus ... 4846
Type ... 4847
CiBands ... 4848
MaPeriod ... 4850
MaShift ... 4851
Deviation ... 4852
Applied ... 4853
Create ... 4854
Base ... 4855
Upper ... 4856
Lower ... 4857
Type ... 4858
CiEnvelopes ... 4859
MaPeriod ... 4861
MaShift ... 4862
MaMethod ... 4863
Deviation ... 4864
Applied ... 4865
Create ... 4866
Upper ... 4867
Lower ... 4868
Type ... 4869

© 2000-2024, MetaQuotes Ltd.

CiIchimoku 4870
TenkanSenPeriod 4872
KijunSenPeriod 4873
SenkouSpanBPeriod 4874
Create 4875
TenkanSen 4876
KijunSen 4877
SenkouSpanA 4878
SenkouSpanB 4879
ChinkouSpan 4880
Type 4881
CiMA 4882
MaPeriod 4884
MaShift 4885
MaMethod 4886
Applied 4887
Create 4888
Main 4889
Type 4890
CiSAR 4891
SarStep 4893
Maximum 4894
Create 4895
Main 4896
Type 4897
CiStdDev 4898
MaPeriod 4900
MaShift 4901
MaMethod 4902
Applied 4903
Create 4904
Main 4905
Type 4906
CiDEMA 4907
MaPeriod 4909
IndShift 4910
Applied 4911
Create 4912
Main 4913
Type 4914
CiTEMA 4915
MaPeriod 4917
IndShift 4918
Applied 4919
Create 4920
Main 4921
Type 4922
CiFrAMA 4923
MaPeriod 4925
IndShift 4926
Applied 4927
Create 4928
Main 4929
Type 4930
CiAMA 4931
MaPeriod 4933
FastEmaPeriod 4934
SlowEmaPeriod 4935

© 2000-2024, MetaQuotes Ltd.

IndShift 4936
Applied 4937
Create 4938
Main 4939
Type 4940
CiVIDyA 4941
CmoPeriod 4943
EmaPeriod 4944
IndShift 4945
Applied 4946
Create 4947
Main 4948
Type 4949
Oscillateurs 4950
CiATR 4951
MaPeriod 4953
Create 4954
Main 4955
Type 4956
CiBearsPower 4957
MaPeriod 4959
Create 4960
Main 4961
Type 4962
CiBullsPower 4963
MaPeriod 4965
Create 4966
Main 4967
Type 4968
CiCCI 4969
MaPeriod 4971
Applied 4972
Create 4973
Main 4974
Type 4975
CiChaikin 4976
FastMaPeriod 4978
SlowMaPeriod 4979
MaMethod 4980
Applied 4981
Create 4982
Main 4983
Type 4984
CiDeMarker 4985
MaPeriod 4987
Create 4988
Main 4989
Type 4990
CiForce 4991
MaPeriod 4993
MaMethod 4994
Applied 4995
Create 4996
Main 4997
Type 4998
CiMACD 4999
FastEmaPeriod 5001
SlowEmaPeriod 5002

© 2000-2024, MetaQuotes Ltd.

SignalPeriod ... 5003
Applied ... 5004
Create ... 5005
Main ... 5006
Signal ... 5007
Type ... 5008
CiMomentum ... 5009
MaPeriod ... 5011
Applied ... 5012
Create ... 5013
Main ... 5014
Type ... 5015
CiOsMA ... 5016
FastEmaPeriod ... 5018
SlowEmaPeriod ... 5019
SignalPeriod ... 5020
Applied ... 5021
Create ... 5022
Main ... 5023
Type ... 5024
CiRSI ... 5025
MaPeriod ... 5027
Applied ... 5028
Create ... 5029
Main ... 5030
Type ... 5031
CiRVI ... 5032
MaPeriod ... 5034
Create ... 5035
Main ... 5036
Signal ... 5037
Type ... 5038
CiStochastic ... 5039
Kperiod ... 5041
Dperiod ... 5042
Slowing ... 5043
MaMethod ... 5044
PriceField ... 5045
Create ... 5046
Main ... 5047
Signal ... 5048
Type ... 5049
CiTriX ... 5050
MaPeriod ... 5052
Applied ... 5053
Create ... 5054
Main ... 5055
Type ... 5056
CiWPR ... 5057
CalcPeriod ... 5059
Create ... 5060
Main ... 5061
Type ... 5062
Indicateurs de Volumes ... 5063
CiAD ... 5064
Applied ... 5066
Create ... 5067
Main ... 5068

© 2000-2024, MetaQuotes Ltd.

Type 5069
CiMFI 5070
MaPeriod 5072
Applied 5073
Create 5074
Main 5075
Type 5076
CiOBV 5077
Applied 5079
Create 5080
Main 5081
Type 5082
CiVolumes 5083
Applied 5085
Create 5086
Main 5087
Type 5088
Indicateurs de Bill Williams 5089
CiAC 5090
Create 5092
Main 5093
Type 5094
CiAlligator 5095
JawPeriod 5097
JawShift 5098
TeethPeriod 5099
TeethShift 5100
LipsPeriod 5101
LipsShift 5102
MaMethod 5103
Applied 5104
Create 5105
Jaw 5106
Teeth 5107
Lips 5108
Type 5109
CiAO 5110
Create 5112
Main 5113
Type 5114
CiFractals 5115
Create 5117
Upper 5118
Lower 5119
Type 5120
CiGator 5121
JawPeriod 5123
JawShift 5124
TeethPeriod 5125
TeethShift 5126
LipsPeriod 5127
LipsShift 5128
MaMethod 5129
Applied 5130
Create 5131
Upper 5132
Lower 5133
Type 5134

© 2000-2024, MetaQuotes Ltd.

CiBWMFI 5135
Applied 5137
Create 5138
Main 5139
Type 5140
Indicateurs personnalisés 5141
NumBuffers 5142
NumParams 5143
ParamType 5144
ParamLong 5145
ParamDouble 5146
ParamString 5147
Type 5148
Classes pour le Trading 5149
CAccountInfo 5150
Login 5152
TradeMode 5153
TradeModeDescription 5154
Leverage 5155
StopoutMode 5156
StopoutModeDescription 5157
MarginMode 5158
MarginModeDescription 5159
TradeAllowed 5160
TradeExpert 5161
LimitOrders 5162
Balance 5163
Credit 5164
Profit 5165
Equity 5166
Margin 5167
FreeMargin 5168
MarginLevel 5169
MarginCall 5170
MarginStopOut 5171
Name 5172
Server 5173
Currency 5174
Company 5175
InfoInteger 5176
InfoDouble 5177
InfoString 5178
OrderProfitCheck 5179
MarginCheck 5180
FreeMarginCheck 5181
MaxLotCheck 5182
CSymbolInfo 5183
Refresh 5187
RefreshRates 5188
Name 5189
Select 5190
IsSynchronized 5191
Volume 5192
VolumeHigh 5193
VolumeLow 5194
Time 5195
Spread 5196
SpreadFloat 5197

© 2000-2024, MetaQuotes Ltd.

TicksBookDepth 5198
StopsLevel 5199
FreezeLevel 5200
Bid 5201
BidHigh 5202
BidLow 5203
Ask 5204
AskHigh 5205
AskLow 5206
Last 5207
LastHigh 5208
LastLow 5209
TradeCalcMode 5210
TradeCalcModeDescription 5211
TradeMode 5212
TradeModeDescription 5213
TradeExecution 5214
TradeExecutionDescription 5215
SwapMode 5216
SwapModeDescription 5217
SwapRollover3days 5218
SwapRollover3daysDescription 5219
MarginInitial 5220
MarginMaintenance 5221
MarginLong 5222
MarginShort 5223
MarginLimit 5224
MarginStop 5225
MarginStopLimit 5226
TradeTimeFlags 5227
TradeFillFlags 5228
Digits 5229
Point 5230
TickValue 5231
TickValueProfit 5232
TickValueLoss 5233
TickSize 5234
ContractSize 5235
LotsMin 5236
LotsMax 5237
LotsStep 5238
LotsLimit 5239
SwapLong 5240
SwapShort 5241
CurrencyBase 5242
CurrencyProfit 5243
CurrencyMargin 5244
Bank 5245
Description 5246
Path 5247
SessionDeals 5248
SessionBuyOrders 5249
SessionSellOrders 5250
SessionTurnover 5251
SessionInterest 5252
SessionBuyOrdersVolume 5253
SessionSellOrdersVolume 5254
SessionOpen 5255

© 2000-2024, MetaQuotes Ltd.

SessionClose ... 5256
SessionAW ... 5257
SessionPriceSettlement ... 5258
SessionPriceLimitMin ... 5259
SessionPriceLimitMax ... 5260
InfoInteger ... 5261
InfoDouble ... 5262
InfoString ... 5263
NormalizePrice ... 5264
COrderInfo ... 5265
Ticket ... 5268
TimeSetup ... 5269
TimeSetupMsc ... 5270
OrderType ... 5271
TypeDescription ... 5272
Etat ... 5273
StateDescription ... 5274
TimeExpiration ... 5275
TimeDone ... 5276
TimeDoneMsc ... 5277
TypeFilling ... 5278
TypeFillingDescription ... 5279
TypeTime ... 5280
TypeTimeDescription ... 5281
Magic ... 5282
PositionId ... 5283
VolumeInitial ... 5284
VolumeCurrent ... 5285
PriceOpen ... 5286
StopLoss ... 5287
TakeProfit ... 5288
PriceCurrent ... 5289
PriceStopLimit ... 5290
Symbol ... 5291
Comment ... 5292
InfoInteger ... 5293
InfoDouble ... 5294
InfoString ... 5295
StoreState ... 5296
CheckState ... 5297
Select ... 5298
SelectByIndex ... 5299
CHistoryOrderInfo ... 5300
TimeSetup ... 5302
TimeSetupMsc ... 5303
OrderType ... 5304
TypeDescription ... 5305
Etat ... 5306
StateDescription ... 5307
TimeExpiration ... 5308
TimeDone ... 5309
TimeDoneMsc ... 5310
TypeFilling ... 5311
TypeFillingDescription ... 5312
TypeTime ... 5313
TypeTimeDescription ... 5314
Magic ... 5315
PositionId ... 5316

© 2000-2024, MetaQuotes Ltd.

VolumeInitial ... 5317
VolumeCurrent ... 5318
PriceOpen ... 5319
StopLoss ... 5320
TakeProfit ... 5321
PriceCurrent ... 5322
PriceStopLimit ... 5323
Symbol ... 5324
Comment ... 5325
InfoInteger ... 5326
InfoDouble ... 5327
InfoString ... 5328
Ticket ... 5329
SelectByIndex ... 5330
CPositionInfo ... 5331
Time ... 5333
TimeMsc ... 5334
TimeUpdate ... 5335
TimeUpdateMsc ... 5336
PositionType ... 5337
TypeDescription ... 5338
Magic ... 5339
Identifier ... 5340
Volume ... 5341
PriceOpen ... 5342
StopLoss ... 5343
TakeProfit ... 5344
PriceCurrent ... 5345
Commission ... 5346
Swap ... 5347
Profit ... 5348
Symbol ... 5349
Comment ... 5350
InfoInteger ... 5351
InfoDouble ... 5352
InfoString ... 5353
Select ... 5354
SelectByIndex ... 5355
SelectByMagic ... 5356
SelectByTicket ... 5357
StoreState ... 5358
CheckState ... 5359
CDealInfo ... 5360
Order ... 5362
Time ... 5363
TimeMsc ... 5364
DealType ... 5365
TypeDescription ... 5366
Entry ... 5367
EntryDescription ... 5368
Magic ... 5369
PositionId ... 5370
Volume ... 5371
Price ... 5372
Commision ... 5373
Swap ... 5374
Profit ... 5375
Symbol ... 5376

© 2000-2024, MetaQuotes Ltd.

Comment ... 5377
InfoInteger ... 5378
InfoDouble ... 5379
InfoString ... 5380
Ticket ... 5381
SelectByIndex ... 5382
CTrade ... 5383
LogLevel ... 5387
SetExpertMagicNumber ... 5388
SetDeviationInPoints ... 5389
SetTypeFilling ... 5390
SetTypeFillingBySymbol ... 5391
SetAsyncMode ... 5392
SetMarginMode ... 5393
OrderOpen ... 5394
OrderModify ... 5396
OrderDelete ... 5397
PositionOpen ... 5398
PositionModify ... 5399
PositionClose ... 5400
PositionClosePartial ... 5401
PositionCloseBy ... 5403
Buy ... 5404
Sell ... 5405
BuyLimit ... 5406
BuyStop ... 5407
SellLimit ... 5408
SellStop ... 5409
Request ... 5410
RequestAction ... 5411
RequestActionDescription ... 5412
RequestMagic ... 5413
RequestOrder ... 5414
RequestSymbol ... 5415
RequestVolume ... 5416
RequestPrice ... 5417
RequestStopLimit ... 5418
RequestSL ... 5419
RequestTP ... 5420
RequestDeviation ... 5421
RequestType ... 5422
RequestTypeDescription ... 5423
RequestTypeFilling ... 5424
RequestTypeFillingDescription ... 5425
RequestTypeTime ... 5426
RequestTypeTimeDescription ... 5427
RequestExpiration ... 5428
RequestComment ... 5429
RequestPosition ... 5430
RequestPositionBy ... 5431
Result ... 5432
ResultRetcode ... 5433
ResultRetcodeDescription ... 5434
ResultDeal ... 5435
ResultOrder ... 5436
ResultVolume ... 5437
ResultPrice ... 5438
ResultBid ... 5439

© 2000-2024, MetaQuotes Ltd.

ResultAsk ... 5440
ResultComment ... 5441
CheckResult ... 5442
CheckResultRetcode ... 5443
CheckResultRetcodeDescription ... 5444
CheckResultBalance ... 5445
CheckResultEquity ... 5446
CheckResultProfit ... 5447
CheckResultMargin ... 5448
CheckResultMarginFree ... 5449
CheckResultMarginLevel ... 5450
CheckResultComment ... 5451
PrintRequest ... 5452
PrintResult ... 5453
FormatRequest ... 5454
FormatRequestResult ... 5455
CTerminalInfo ... 5456
Build ... 5458
IsConnected ... 5459
IsDLLsAllowed ... 5460
IsTradeAllowed ... 5461
IsEmailEnabled ... 5462
IsFtpEnabled ... 5463
MaxBars ... 5464
CodePage ... 5465
CPUCores ... 5466
MemoryPhysical ... 5467
MemoryTotal ... 5468
MemoryAvailable ... 5469
MemoryUsed ... 5470
IsX64 ... 5471
OpenCLSupport ... 5472
DiskSpace ... 5473
Language ... 5474
Name ... 5475
Company ... 5476
Path ... 5477
DataPath ... 5478
CommonDataPath ... 5479
InfoInteger ... 5480
InfoString ... 5481
Modules de Stratégies ... 5482
Classes de base des Expert Advisors ... 5485
CExpertBase ... 5486
InitPhase ... 5489
TrendType ... 5490
UsedSeries ... 5491
EveryTick ... 5492
Open ... 5493
High ... 5494
Low ... 5495
Close ... 5496
Spread ... 5497
Time ... 5498
TickVolume ... 5499
RealVolume ... 5500
Init ... 5501
Symbol ... 5502

© 2000-2024, MetaQuotes Ltd.

Period ... 5503
Magic ... 5504
ValidationSettings ... 5505
SetPriceSeries ... 5506
SetOtherSeries ... 5507
InitIndicators ... 5508
InitOpen ... 5509
InitHigh ... 5510
InitLow ... 5511
InitClose ... 5512
InitSpread ... 5513
InitTime ... 5514
InitTickVolume ... 5515
InitRealVolume ... 5516
PriceLevelUnit ... 5517
StartIndex ... 5518
CompareMagic ... 5519
CExpert ... 5520
Init ... 5525
Magic ... 5526
InitSignal ... 5527
InitTrailing ... 5528
InitMoney ... 5529
InitTrade ... 5530
Deinit ... 5531
OnTickProcess ... 5532
OnTradeProcess ... 5533
OnTimerProcess ... 5534
OnChartEventProcess ... 5535
OnBookEventProcess ... 5536
MaxOrders ... 5537
Signal ... 5538
ValidationSettings ... 5539
InitIndicators ... 5540
OnTick ... 5541
OnTrade ... 5542
OnTimer ... 5543
OnChartEvent ... 5544
OnBookEvent ... 5545
InitParameters ... 5546
DeinitTrade ... 5547
DeinitSignal ... 5548
DeinitTrailing ... 5549
DeinitMoney ... 5550
DeinitIndicators ... 5551
Refresh ... 5552
Processing ... 5553
SelectPosition ... 5555
CheckOpen ... 5556
CheckOpenLong ... 5557
CheckOpenShort ... 5558
OpenLong ... 5559
OpenShort ... 5560
CheckReverse ... 5561
CheckReverseLong ... 5562
CheckReverseShort ... 5563
ReverseLong ... 5564
ReverseShort ... 5565

© 2000-2024, MetaQuotes Ltd.

CheckClose ... 5566
CheckCloseLong ... 5567
CheckCloseShort ... 5568
CloseAll ... 5569
Close ... 5570
CloseLong ... 5571
CloseShort ... 5572
CheckTrailingStop ... 5573
CheckTrailingStopLong ... 5574
CheckTrailingStopShort ... 5575
TrailingStopLong ... 5576
TrailingStopShort ... 5577
CheckTrailingOrderLong ... 5578
CheckTrailingOrderShort ... 5579
TrailingOrderLong ... 5580
TrailingOrderShort ... 5581
CheckDeleteOrderLong ... 5582
CheckDeleteOrderShort ... 5583
DeleteOrders ... 5584
DeleteOrder ... 5585
DeleteOrderLong ... 5586
DeleteOrderShort ... 5587
LotOpenLong ... 5588
LotOpenShort ... 5589
LotReverse ... 5590
PrepareHistoryDate ... 5591
HistoryPoint ... 5592
CheckTradeState ... 5593
WaitEvent ... 5594
NoWaitEvent ... 5595
TradeEventPositionStopTake ... 5596
TradeEventOrderTriggered ... 5597
TradeEventPositionOpened ... 5598
TradeEventPositionVolumeChanged ... 5599
TradeEventPositionModified ... 5600
TradeEventPositionClosed ... 5601
TradeEventOrderPlaced ... 5602
TradeEventOrderModified ... 5603
TradeEventOrderDeleted ... 5604
TradeEventNotIdentified ... 5605
TimeframeAdd ... 5606
TimeframesFlags ... 5607
CExpertSignal ... 5608
BasePrice ... 5611
UsedSeries ... 5612
Weight ... 5613
PatternsUsage ... 5614
General ... 5615
Ignore ... 5616
Invert ... 5617
ThresholdOpen ... 5618
ThresholdClose ... 5619
PriceLevel ... 5620
StopLevel ... 5621
TakeLevel ... 5622
Expiration ... 5623
Magic ... 5624
ValidationSettings ... 5625

© 2000-2024, MetaQuotes Ltd.

InitIndicators ... 5626
AddFilter ... 5627
CheckOpenLong ... 5628
CheckOpenShort ... 5629
OpenLongParams ... 5630
OpenShortParams ... 5631
CheckCloseLong ... 5632
CheckCloseShort ... 5633
CloseLongParams ... 5634
CloseShortParams ... 5635
CheckReverseLong ... 5636
CheckReverseShort ... 5637
CheckTrailingOrderLong ... 5638
CheckTrailingOrderShort ... 5639
LongCondition ... 5640
ShortCondition ... 5641
Direction ... 5642
CExpertTrailing ... 5643
CheckTrailingStopLong ... 5645
CheckTrailingStopShort ... 5646
CExpertMoney ... 5647
Percent ... 5649
ValidationSettings ... 5650
CheckOpenLong ... 5651
CheckOpenShort ... 5652
CheckReverse ... 5653
CheckClose ... 5654
Modules des Signaux de Trading ... 5655
Signaux de l'indicateur Oscillateur d'Accélération ... 5658
Signaux de l'Indicateur Moyenne Mobile Adaptative ... 5661
Signaux de l'indicateur Awesome Oscillator ... 5665
Signaux de l'indicateur Oscillateur Bears Power ... 5669
Signaux de l'indicateur Oscillateur Bulls Power ... 5671
Signaux de l'indicateur Oscillateur Commodity Channel Index ... 5673
Signaux de l'indicateur Oscillateur DeMarker ... 5677
Signaux de l'indicateur Double Exponential Moving Average ... 5681
Signaux de l'indicateur Envelopes ... 5685
Signaux de l'indicateur Fractal Adaptive Moving Average ... 5688
Signaux du Filtre Horaire Intraday ... 5692
Signaux de l'oscillateur MACD ... 5694
Signaux de l'indicateur Moving Average ... 5700
Signaux de l'indicateur Parabolic SAR ... 5704
Signaux de l'oscillateur Relative Strength Index ... 5706
Signaux de l'oscillateur Relative Vigor Index ... 5712
Signaux de l'oscillateur Stochastic ... 5714
Signaux de l'oscillateur Triple Exponential Average ... 5719
Signaux de l'indicateur Triple Exponential Moving Average ... 5723
Signaux de l'oscillatuer Williams Percent Range ... 5727
Classes de Trailing Stop (Stops Suiveurs) ... 5730
CTrailingFixedPips ... 5731
StopLevel ... 5733
ProfitLevel ... 5734
ValidationSettings ... 5735
CheckTrailingStopLong ... 5736
CheckTrailingStopShort ... 5737
CTrailingMA ... 5738
Period ... 5740
Shift ... 5741

© 2000-2024, MetaQuotes Ltd.

Method ... 5742
Applied ... 5743
InitIndicators ... 5744
ValidationSettings ... 5745
CheckTrailingStopLong ... 5746
CheckTrailingStopShort ... 5747
CTrailingNone ... 5748
CheckTrailingStopLong ... 5749
CheckTrailingStopShort ... 5750
CTrailingPSAR ... 5751
Step ... 5753
Maximum ... 5754
InitIndicators ... 5755
CheckTrailingStopLong ... 5756
CheckTrailingStopShort ... 5757
Classes de Money Management ... 5758
CMoneyFixedLot ... 5759
Lots ... 5761
ValidationSettings ... 5762
CheckOpenLong ... 5763
CheckOpenShort ... 5764
CMoneyFixedMargin ... 5765
CheckOpenLong ... 5766
CheckOpenShort ... 5767
CMoneyFixedRisk ... 5768
CheckOpenLong ... 5769
CheckOpenShort ... 5770
CMoneyNone ... 5771
ValidationSettings ... 5772
CheckOpenLong ... 5773
CheckOpenShort ... 5774
CMoneySizeOptimized ... 5775
DecreaseFactor ... 5777
ValidationSettings ... 5778
CheckOpenLong ... 5779
CheckOpenShort ... 5780
Panneaux et Dialogues ... 5781
CRect ... 5783
Left ... 5784
Top ... 5785
Right ... 5786
Bottom ... 5787
Width ... 5788
Height ... 5789
SetBound ... 5790
Move ... 5791
Shift ... 5792
Contains ... 5793
Format ... 5794
CDateTime ... 5795
MonthName ... 5797
ShortMonthName ... 5798
DayName ... 5799
ShortDayName ... 5800
DaysInMonth ... 5801
DateTime ... 5802
Date ... 5803
Time ... 5804

© 2000-2024, MetaQuotes Ltd.

Sec ... 5805
Min ... 5806
Hour ... 5807
Day ... 5808
Mon ... 5809
Year ... 5810
SecDec ... 5811
SecInc ... 5812
MinDec ... 5813
MinInc ... 5814
HourDec ... 5815
HourInc ... 5816
DayDec ... 5817
DayInc ... 5818
MonDec ... 5819
MonInc ... 5820
YearDec ... 5821
YearInc ... 5822
CWnd ... 5823
Create ... 5827
Destroy ... 5828
OnEvent ... 5829
OnMouseEvent ... 5830
Name ... 5831
ControlsTotal ... 5832
Control ... 5833
ControlFind ... 5834
Rect ... 5835
Left ... 5836
Top ... 5837
Right ... 5838
Bottom ... 5839
Width ... 5840
Height ... 5841
Move ... 5842
Shift ... 5843
Resize ... 5844
Contains ... 5845
Alignment ... 5846
Align ... 5847
Id ... 5848
IsEnabled ... 5849
Enable ... 5850
Disable ... 5851
IsVisible ... 5852
Visible ... 5853
Show ... 5854
Hide ... 5855
IsActive ... 5856
Activate ... 5857
Deactivate ... 5858
StateFlags ... 5859
StateFlagsSet ... 5860
StateFlagsReset ... 5861
PropFlags ... 5862
PropFlagsSet ... 5863
PropFlagsReset ... 5864
MouseX ... 5865

© 2000-2024, MetaQuotes Ltd.

MouseY 5866
MouseFlags 5867
MouseFocusKill 5868
OnCreate 5869
OnDestroy 5870
OnMove 5871
OnResize 5872
OnEnable 5873
OnDisable 5874
OnShow 5875
OnHide 5876
OnActivate 5877
OnDeactivate 5878
OnClick 5879
OnChange 5880
OnMouseDown 5881
OnMouseUp 5882
OnDragStart 5883
OnDragProcess 5884
OnDragEnd 5885
DragObjectCreate 5886
DragObjectDestroy 5887
CWndObj 5888
OnEvent 5890
Texte 5891
Color 5892
ColorBackground 5893
ColorBorder 5894
Font 5895
FontSize 5896
ZOrder 5897
OnObjectCreate 5898
OnObjectChange 5899
OnObjectDelete 5900
OnObjectDrag 5901
OnSetText 5902
OnSetColor 5903
OnSetColorBackground 5904
OnSetFont 5905
OnSetFontSize 5906
OnSetZOrder 5907
OnDestroy 5908
OnChange 5909
CWndContainer 5910
Destroy 5912
OnEvent 5913
OnMouseEvent 5914
ControlsTotal 5915
Control 5916
ControlFind 5917
Add 5918
Delete 5919
Move 5920
Shift 5921
Id 5922
Enable 5923
Disable 5924
Show 5925

© 2000-2024, MetaQuotes Ltd.

Hide ... 5926
MouseFocusKill ... 5927
Save ... 5928
Load ... 5929
OnResize ... 5930
OnActivate ... 5931
OnDeactivate ... 5932
CLabel ... 5933
Create ... 5938
OnSetText ... 5939
OnSetColor ... 5940
OnSetFont ... 5941
OnSetFontSize ... 5942
OnCreate ... 5943
OnShow ... 5944
OnHide ... 5945
OnMove ... 5946
CBmpButton ... 5947
Create ... 5953
Border ... 5954
BmpNames ... 5955
BmpOffName ... 5956
BmpOnName ... 5957
BmpPassiveName ... 5958
BmpActiveName ... 5959
Pressed ... 5960
Locking ... 5961
OnSetZOrder ... 5962
OnCreate ... 5963
OnShow ... 5964
OnHide ... 5965
OnMove ... 5966
OnChange ... 5967
OnActivate ... 5968
OnDeactivate ... 5969
OnMouseDown ... 5970
OnMouseUp ... 5971
CButton ... 5972
Create ... 5979
Pressed ... 5980
Locking ... 5981
OnSetText ... 5982
OnSetColor ... 5983
OnSetColorBackground ... 5984
OnSetColorBorder ... 5985
OnSetFont ... 5986
OnSetFontSize ... 5987
OnCreate ... 5988
OnShow ... 5989
OnHide ... 5990
OnMove ... 5991
OnResize ... 5992
OnMouseDown ... 5993
OnMouseUp ... 5994
CEdit ... 5995
Create ... 6000
ReadOnly ... 6001
TextAlign ... 6002

© 2000-2024, MetaQuotes Ltd.

OnObjectEndEdit ... 6003
OnSetText ... 6004
OnSetColor ... 6005
OnSetColorBackground ... 6006
OnSetColorBorder ... 6007
OnSetFont ... 6008
OnSetFontSize ... 6009
OnSetZOrder ... 6010
OnCreate ... 6011
OnShow ... 6012
OnHide ... 6013
OnMove ... 6014
OnResize ... 6015
OnChange ... 6016
OnClick ... 6017
CPanel ... 6018
Create ... 6023
BorderType ... 6024
OnSetText ... 6025
OnSetColorBackground ... 6026
OnSetColorBorder ... 6027
OnCreate ... 6028
OnShow ... 6029
OnHide ... 6030
OnMove ... 6031
OnResize ... 6032
OnChange ... 6033
CPicture ... 6034
Create ... 6039
Border ... 6040
BmpName ... 6041
OnCreate ... 6042
OnShow ... 6043
OnHide ... 6044
OnMove ... 6045
OnChange ... 6046
CScroll ... 6047
Create ... 6049
OnEvent ... 6050
MinPos ... 6051
MaxPos ... 6052
CurrPos ... 6053
CreateBack ... 6054
CreateInc ... 6055
CreateDec ... 6056
CreateThumb ... 6057
OnClickInc ... 6058
OnClickDec ... 6059
OnShow ... 6060
OnHide ... 6061
OnChangePos ... 6062
OnThumbDragStart ... 6063
OnThumbDragProcess ... 6064
OnThumbDragEnd ... 6065
CalcPos ... 6066
CScrollV ... 6067
CreateInc ... 6073
CreateDec ... 6074

© 2000-2024, MetaQuotes Ltd.

CreateThumb ... 6075
OnResize ... 6076
OnChangePos ... 6077
OnThumbDragStart ... 6078
OnThumbDragProcess ... 6079
OnThumbDragEnd ... 6080
CalcPos ... 6081
CScrollH ... 6082
CreateInc ... 6088
CreateDec ... 6089
CreateThumb ... 6090
OnResize ... 6091
OnChangePos ... 6092
OnThumbDragStart ... 6093
OnThumbDragProcess ... 6094
OnThumbDragEnd ... 6095
CalcPos ... 6096
CWndClient ... 6097
Create ... 6100
OnEvent ... 6101
ColorBackground ... 6102
ColorBorder ... 6103
BorderType ... 6104
VScrolled ... 6105
HScrolled ... 6106
CreateBack ... 6107
CreateScrollV ... 6108
CreateScrollH ... 6109
OnResize ... 6110
OnVScrollShow ... 6111
OnVScrollHide ... 6112
OnHScrollShow ... 6113
OnHScrollHide ... 6114
OnScrollLineDown ... 6115
OnScrollLineUp ... 6116
OnScrollLineLeft ... 6117
OnScrollLineRight ... 6118
Rebound ... 6119
CListView ... 6120
Create ... 6126
OnEvent ... 6127
TotalView ... 6128
AddItem ... 6129
Select ... 6130
SelectByText ... 6131
SelectByValue ... 6132
Value ... 6133
CreateRow ... 6134
OnResize ... 6135
OnVScrollShow ... 6136
OnVScrollHide ... 6137
OnScrollLineDown ... 6138
OnScrollLineUp ... 6139
OnItemClick ... 6140
Redraw ... 6141
RowState ... 6142
CheckView ... 6143
CComboBox ... 6144

© 2000-2024, MetaQuotes Ltd.

Create ... 6150
OnEvent ... 6151
AddItem ... 6152
ListViewItems ... 6153
Select ... 6154
SelectByText ... 6155
SelectByValue ... 6156
Value ... 6157
CreateEdit ... 6158
CreateButton ... 6159
CreateList ... 6160
OnClickEdit ... 6161
OnClickButton ... 6162
OnChangeList ... 6163
ListShow ... 6164
ListHide ... 6165
CCheckBox ... 6166
Create ... 6172
OnEvent ... 6173
Texte ... 6174
Color ... 6175
Checked ... 6176
Value ... 6177
CreateButton ... 6178
CreateLabel ... 6179
OnClickButton ... 6180
OnClickLabel ... 6181
CCheckGroup ... 6182
Create ... 6188
OnEvent ... 6189
AddItem ... 6190
Value ... 6191
CreateButton ... 6192
OnVScrollShow ... 6193
OnVScrollHide ... 6194
OnScrollLineDown ... 6195
OnScrollLineUp ... 6196
OnChangeItem ... 6197
Redraw ... 6198
RowState ... 6199
CRadioButton ... 6200
Create ... 6202
OnEvent ... 6203
Texte ... 6204
Color ... 6205
Etat ... 6206
CreateButton ... 6207
CreateLabel ... 6208
OnClickButton ... 6209
OnClickLabel ... 6210
CRadioGroup ... 6211
Create ... 6217
OnEvent ... 6218
AddItem ... 6219
Value ... 6220
CreateButton ... 6221
OnVScrollShow ... 6222
OnVScrollHide ... 6223

© 2000-2024, MetaQuotes Ltd.

OnScrollLineDown ... 6224
OnScrollLineUp ... 6225
OnChangeItem ... 6226
Redraw ... 6227
RowState ... 6228
Select ... 6229
CSpinEdit ... 6230
Create ... 6236
OnEvent ... 6237
MinValue ... 6238
MaxValue ... 6239
Value ... 6240
CreateEdit ... 6241
CreateInc ... 6242
CreateDec ... 6243
OnClickInc ... 6244
OnClickDec ... 6245
OnChangeValue ... 6246
CDialog ... 6247
Create ... 6250
OnEvent ... 6251
Caption ... 6252
Add ... 6253
CreateWhiteBorder ... 6254
CreateBackground ... 6255
CreateCaption ... 6256
CreateButtonClose ... 6257
CreateClientArea ... 6258
OnClickCaption ... 6259
OnClickButtonClose ... 6260
ClientAreaVisible ... 6261
ClientAreaLeft ... 6262
ClientAreaTop ... 6263
ClientAreaRight ... 6264
ClientAreaBottom ... 6265
ClientAreaWidth ... 6266
ClientAreaHeight ... 6267
OnDialogDragStart ... 6268
OnDialogDragProcess ... 6269
OnDialogDragEnd ... 6270
CAppDialog ... 6271
Create ... 6274
Destroy ... 6275
OnEvent ... 6276
Run ... 6277
ChartEvent ... 6278
Minimized ... 6279
IniFileSave ... 6280
IniFileLoad ... 6281
IniFileName ... 6282
IniFileExt ... 6283
CreateCommon ... 6284
CreateExpert ... 6285
CreateIndicator ... 6286
CreateButtonMinMax ... 6287
OnClickButtonClose ... 6288
OnClickButtonMinMax ... 6289
OnAnotherApplicationClose ... 6290

© 2000-2024, MetaQuotes Ltd.

Rebound 6291
Minimize 6292
Maximize 6293
CreateInstanceId 6294
ProgramName 6295
SubwinOff 6296
36 Migration depuis MQL4 6297
37 Liste des fonctions MQL5 6301
38 Liste des constantes MQL5 6337

© 2000-2024, MetaQuotes Ltd.

Référence MQL5

Le MetaQuotes Language 5 (MQL5) est un langage haut niveau conçu pour développer des indicateurs techniques, des robots de trading et des applications utilitaires qui automatisent le trading financier. Le MQL5 a été développé par MetaQuotes pour leurs plateformes de trading. La syntaxe du langage est très proche du C++, permettant aux programmeurs de développer des applications en Programmation Orientée Objet (POO ou OOP en anglais).

En plus du langage MQL5, le package de la plateforme de trading inclut aussi l'IDE MetaEditor avec des outils avancés d'écriture du code, tels que les templates, les snippets, le déboguage, le profilage et des outils d'auto-complétion, ainsi que le MQL5 Storage permettant les versions des fichiers.

Le support du langage est disponible sur le site web de la MQL5.community, qui contient une énorme base de code CodeBase gratuite et une pléthode d'articles. Ces articles couvrent tous les aspects du trading moderne, incluant les réseaux neuronaux, les statistiques et l'analyse, le trading à haute fréquence, l'arbitrage, le test et l'optimisation de stratégies de trading, l'utilisation de robots de trading automatiques, et plus.

Les traders et les développeurs de programmes MQL5 peuvent discuter sur le forum, commander et développer des applications en utilisant le service Freelance, ainsi qu'acheter et vendre des programmes protégés dans le Market d'applications de trading automatisées.

Le langage MQL5 fournit des fonctions de trading spécialisées et des gestionnaures d'évènements prédéfinis pour aider les programmeurs à développer des Expert Advisors (EA), qui contrôle automatiquement les process de trading suivant des règles de trading spécifiques. En plus des EA, MQL5 permet de développer des indicateurs techniques, des scripts et des bibliothèques.

Cette référence du langage MQL5 contient les fonctions, les opérations, les mots réservés et d'autres constructions du langage, divisées en catégories. La référence fournit également les descriptions des classes de la Bibliothèque Standard utilisée pour développer des stratégies de trading, des panneaux de contrôle, des graphiques personnalisés et utiliser les fichiers.

De plus, la CodeBase contient la bibliothèque d'analyse numérique ALGLIB, qui peut être utilisée pour résoudre différents problèmes mathématiques.

Types d'Applications MQL5

Les programmes MQL5 sont divisés en cinq types spécialisés selon les tâches automatiques de trading qu'ils implémentent :

- **Expert Advisor** est un système de trading automatisé lié à un graphique. Un Expert Advisor contient des gestionnaires d'évènements pour gérer les évènements prédéfinis qui activent l'exécution des éléments de la stratégie de trading désirée. Par exemple, un évènement d'initialisation et de désinitialisation du programme, de nouveaux ticks, d'évènements du time, de changements dans le Depth of Market, dans le graphique et des évènements personnalisés.
  En plus de calculer les signaux de trading selon les règles implémentées, les Expert Advisors peuvent également exécuter automatiquement des trades et les envoyer directement au serveur de trading. Les Expert Advisors sont stockés dans *<Répertoire_du_terminal>\MQL5\Experts*.
- Un **indicateur personnalisé** est un indicateur technique développé par un utilisateur en plus des indicateurs standards intégrés dans la plateforme de trading. Les indicateurs personnalisés, comme les indicateurs standards, ne peuvent pas trader automatiquement, mais implémentent uniquement

© 2000-2024, MetaQuotes Ltd.

des fonctions d'analyse. Les indicateurs personnalisés peuvent utiliser les valeurs d'autres indicateurs pour les calculs, et peuvent être appelés depuis les Expert Advisors.
Les indicateurs personnalisés sont stockés dans <*Répertoire_du_terminal*>*MQL5**Indicators*.

- Un **script** est un programme pour l'exécution unique d'une action. Contrairement aux Expert Advisors, les scripts ne gèrent aucun évènement à part son activation, son initialisation et sa désinitialisation. Le code d'un script doit contenir la fonction OnStart.
  Les scripts sont stockés dans <*Répertoire_du_terminal*>*MQL5**Scripts*.
- Une **bibliothèque** est un ensemble de fonctions personnalisées. Les bibliothèques sont prévues pour stocker et distribuer les algorithmes communément utilisés de programmes personnalisés.
  Les bibliothèques sont stockées dans <*Répertoire_du_terminal*>*MQL5**Libraries*.
- Un **service** est un programme qui, contrairement aux indicateurs, Expert Advisors et scripts, ne nécessite pas d'être lié à un graphique pour fonctionner. Comme les scripts, les services ne gèrent aucun évènement excepté pour les triggers. Pour lancer un servier, son code doit contenir la fonction OnStart. Les services n'acceptent pas d'évènements autres que Start, mais ils peuvent envoyer des évènements personnalisés aux graphiques grâce à la fonction EventChartCustom. Les services sont situés dans <*Répertoire_du_terminal*>*MQL5**Services*.
- Un **fichier Include** est un texte source de blocs de code les plus fréquemment utilisés dans les programmes personnalisés. Ces fichiers peuvent être inclus dans le code source des Expert Advisors, des scripts, des indicateurs personnalisés et des bibliothèques au moment de la compilation. L'utilisation de fichiers d'include est préférable à l'utilisation de bibliothèques en raison de la charge supplémentaire au moment de l'appel aux fonctions de la bibliothèque.
  Les fichiers d'include peuvent être stockés dans le même répertoire que le fichier appelant. Dans ce cas, la directive #include doit être entre guillemets. Une autre option est de stocker les fichiers d'include dans <*Répertoire_du_terminal*>*MQL5**Include*. Dans ce cas, #include doit être utilisé avec des chevrons.

© 2000-2024, MetaQuotes Ltd

Les principes du langage

Le langage MetaQuotes Language 5 (MQL5) est le langage orientée objet de programmation de haut niveau et destinée à la rédaction des stratégies automatiques commerciales, des indicateurs techniques d' utilisateurs pour l'analyse des différents marchés financiers. Elle permet non seulement d'écrire une variété de systèmes experts, conçus pour fonctionner en temps réel, mais aussi créer leurs propres outils graphiques pour aider à prendre des décisions commerciales.

MQL5 est basée sur la conception du langage de programmation C++ largement répandu, en comparaison de MQL4 le nouveau langage a maintenant des énumérations, des structures, des classes et le traitement des événements. En augmentant le nombre des types insérés principaux, l'interaction des programmes exécutés dans MQL5 avec d'autres applications par l'intermédiaire de dll est facilitée au maximum. La syntaxe MQL5 est similaire à la syntaxe du langage C++ et ça permet de lui transférer des programmes des langages de programmation moderne.

Pour vous aider à étudier le langage MQL5, tous les thèmes sont regroupés dans les catégories suivantes:

- Syntaxe
- Types de données
- Opérations et expressions
- Opérateurs
- Fonctions
- Variables
- Préprocesseur
- Programmation orientée objet
- Namespaces

© 2000-2024, MetaQuotes Ltd.

Syntaxe

La syntaxe du langage MQL5 pour programmer les stratégies de trading est très similaire au langage de programmation C++, excepté pour certains caractéristiques :

- pas d'arithmétique d'adresse ;
- pas d'opérateur goto ;
- pas de déclaration possible d'énumération anonyme ;
- par d'héritage multiple.

Voir aussi

Enumérations, Structures et Classes, Héritage

© 2000-2024, MetaQuotes Ltd.

Commentaires

Les commentaires multi-lignes commencent avec les symboles /* et se terminent avec */. Ce type de commentaires ne peut pas être imbriqué. Les commentaires sur une seule ligne commencent avec les symboles // et se terminent avec le caractère de fin de ligne, ils peuvent être imbriqués dans des commentaires multi-lignes. Les commentaires sont autorisés partout où les espaces sont autorisés, ils peuvent avoir n'importe quel nombre d'espaces.

Exemples :

```
//--- Commentaire sur une seule ligne
/*  Multi-
   ligne         // Commentaire sur une seule ligne imbriqué
    comment
*/
```

© 2000-2024, MetaQuotes Ltd.

Identifiants

Les identifiants sont utilisés comme noms de variables et de fonctions. La longueur d'un identifiant ne doit pas être supérieure à 63 caractères.

Les caractères autorisés comme identifiant : chiffres de 0 à 9, les lettres latines de a à z en minuscules ou majuscules, reconnues comme des caractères différents, le caractère underscore (_). Le premier caractère ne peut pas être un chiffre.

L'identifiant ne doit pas correspondre à un mot réservé.

Exemples :

```
NAME1 name1 Total_5 Paper
```

Voir aussi

Variables, Fonctions

© 2000-2024, MetaQuotes Ltd.

Mots Réservés

Les identifiants suivants sont des mots réservés, chacun d'eux correspond à une certaine action et ne peut pas être utilisé d'une autre façon :

Types de Données

bool	float	uint
char	int	ulong
class	long	union
color	short	ushort
datetime	string	void
double	struct	
enum	uchar	

Spécificateurs d'Accès

const	private	virtual
delete	protected	
override	public	

Classes Mémoire

extern	input	static

Opérateurs

break	dynamic_cast	operator
case	else	pack
continue	for	return
default	if	sizeof
delete	new	switch
do	offsetof	while

Autre

this	#define	#import
true	#ifdef	#include
false	#ifndef	#property

© 2000-2024, MetaQuotes Ltd.

this	#define	#import
template	#else	group
typename	#endif	namespace

© 2000-2024, MetaQuotes Ltd.

Types de Données

Tous les programmes opèrent avec des données. Les données peuvent être de différents types suivant leur but. Par exemple, les données de type integer sont utilisées pour accéder aux composants d'un tableau. Les prix correspondent aux données avec une double précision et une virgule flottante. Ceci est lié au fait qu'aucun type de donnée spécial pour les prix n'existe en MQL5.

Les données de différents types sont traités différemment. Les entiers sont traités sont le plus rapidement. Pour traiter les données de précision double, un coprocesseur spécial est utilisé. Cependant, en raison de la complexité de la représentation interne des données avec virgule flottante, celles-ci sont traitées plus lentement que les données de type entier.

Les chaînes de caractères prennent le plus de temps de traitement, en raison de l'allocation/réallocation dynamique de la mémoire de l'ordinateur.

Les types de données de base sont :

- entiers (char, short, int, long, uchar, ushort, uint, ulong);
- logiques (bool);
- littéraux (ushort);
- chaînes de caractères (string);
- chiffres à virgule flottante (double, float);
- couleur (color);
- date et heure (datetime);
- énumérations (enum).

Les types de données complexes sont :

- structures;
- classes.

En terme de POO, les types de données complexes sont appelés des types de données abstraits.

Les types *color* et *datetime* ne servent qu'à faciliter la visualisation et la saisie de paramètres définis en dehors - du tableau des propriétés de l'Expert Advisor ou de l'indicateur personnalisé (l'onglet Inputs). Les données de type *color* et *datetime* sont représentés comme des entiers. Les types entiers et à virgule flottante sont appelés des types arithmétiques (numériques).

Seul le le casting de type implicite est utilisé dans les expressions, sauf si un casting explicite est spécifié.

Voir aussi

Conversion de Type

© 2000-2024, MetaQuotes Ltd.

Types Integer

En MQL5, les entiers sont représentés par 11 types différents. Certains types peuvent être utilisés avec d'autres, si cela est requis par la logique du programme, mais dans ce cas, il est nécessaire de se souvenir des règles du casting de type (typecasting).

Le tableau ci-dessous liste les caractéristiques de chaque type. De plus, la dernière colonne donne le type en C++ correspondant à chaque type.

Type	Taille en Octets	Valeur Minimum	Valeur Maximum	Analogie en C++
char	1	-128	127	char
uchar	1	0	255	unsigned char, BYTE
bool	1	0 (false)	1 (true)	bool
short	2	-32 768	32 767	short, wchar_t
ushort	2	0	65 535	unsigned short, WORD
int	4	- 2 147 483 648	2 147 483 647	int
uint	4	0	4 294 967 295	unsigned int, DWORD
color	4	-1	16 777 215	int, COLORREF
long	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	__int64
ulong	8	0	18 446 744 073 709 551 615	unsigned __int64
datetime	8	0 (1970.01.01 0:00:00)	32 535 244 799 (3000.12.31 23:59:59)	__time64_t

Les valeurs de type entier peuvent également être présentées comme des constantes numériques, des littéraux de couleurs, des littéraux de date-heure, des constantes de caractères et des énumérations.

Voir aussi

Fonctions de Conversion, Constantes de Type Numérique

© 2000-2024, MetaQuotes Ltd.

Types Char, Short, Int et Long

char

Le type *char* occupe 1 octet en mémoire (8 bits) et permet d'exprimer 2^8=256 valeurs en représentation binaire. Le type *char* peut contenir des valeurs positives et des valeurs négatives. Les valeurs possibles vont de -128 à 127.

uchar

Le type entier *uchar* occupe également 1 octet en mémoire, comme le type *char*, mais contrairement à lui, *uchar* n'est prévu que pour les valeurs positives. La valeur minimum est zéro, la valeur maximum est 255. La première lettre u dans le nom du type *uchar* est l'abréviation de *unsigned*.

short

La taille du type *short* est de 2 octets (16 bits) et de la même façon, il permet d'exprimer l'intervalle de valeurs égales à 2 puissance 16 : 2^16 = 65 536. Puisque le type *short* est signé, et qu'il contient à la fois des valeurs positives et des valeurs négatives, l'intervalle des valeurs est compris entre -32 768 et 32 767.

ushort

Le type non signé *short* est le type *ushort*, qui a également une taille de 2 octets. La valeur minimum est 0, la valeur maximum est 65 535.

int

La taille du type *int* est de 4 octets (32 bits). La valeur minimale est -2 147 483 648, la valeur maximale est 2 147 483 647.

uint

Le type entier non signé est *uint*. Il occupe 4 octets en mémoire et permet d'exprimer des entiers de 0 à 4 294 967 295.

long

La taille du type *long* est de 8 octets (64 bits). La valeur minimum est -9 223 372 036 854 775 808, la valeur maximum est 9 223 372 036 854 775 807.

ulong

Le type *ulong* occupe également 8 octets et peut stocker des valeurs de 0 à 18 446 744 073 709 551 615.

Exemples :

```
char  ch=12;
```

© 2000-2024, MetaQuotes Ltd.

```
short sh=-5000;
int   in=2445777;
```

Puisque les types entiers non signés ne sont pas conçus pour stocker des valeurs négatives, la tentative d'affecter une valeur négative peut mener à des conséquences inattendues. Le script simple suivant génèrera une boucle infinie :

```
//--- Boucle infinie
void OnStart()
  {
   uchar  u_ch;

   for(char ch=-128;ch<128;ch++)
     {
      u_ch=ch;
      Print("ch = ",ch," u_ch = ",u_ch);
     }
  }
```

La variante correcte est :

```
//--- Variante correcte
void OnStart()
  {
   uchar  u_ch;

   for(char ch=-128;ch<=127;ch++)
     {
      u_ch=ch;
      Print("ch = ",ch," u_ch = ",u_ch);
      if(ch==127) break;
     }
  }
```

Résultat :

```
   ch= -128  u_ch= 128
   ch= -127  u_ch= 129
   ch= -126  u_ch= 130
   ch= -125  u_ch= 131
   ch= -124  u_ch= 132
   ch= -123  u_ch= 133
   ch= -122  u_ch= 134
   ch= -121  u_ch= 135
   ch= -120  u_ch= 136
   ch= -119  u_ch= 137
   ch= -118  u_ch= 138
   ch= -117  u_ch= 139
   ch= -116  u_ch= 140
   ch= -115  u_ch= 141
   ch= -114  u_ch= 142
```

© 2000-2024, MetaQuotes Ltd.

```
   ch= -113  u_ch= 143
   ch= -112  u_ch= 144
   ch= -111  u_ch= 145
    ...
```

Exemples :

```
//--- Les valeurs négatives ne peuvent pas être stockées dans les types non signés
uchar  u_ch=-120;
ushort u_sh=-5000;
uint   u_in=-401280;
```

Héxadecimal : nombres de 0 à 9, lettres de a à f ou A à F pour les valeurs 10 à 15 ; démarre avec 0x ou 0X.

Exemples :

```
0x0A, 0x12, 0X12, 0x2f, 0xA3, 0Xa3, 0X7C7
```

Pour les variables entières, les valeurs peuvent être affectées sous la forme binaire en utilisant le préfixe B. Par exmeple, vous pouvez encoder les heures de travail d'une session de trading dans une variable de type int et utiliser ses informations selon l'algorithme désiré :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- utilise 1 pour les heures travaillées et 0 pour les autres
   int AsianSession   =B'111111111'; // Session asiatique de 0:00 à 9:00
   int EuropeanSession=B'111111111000000000'; // Session européenne 9:00 - 18:00
   int AmericanSession =B'111111110000000000000011'; // Session américaine 16:00 - 02:
//--- dérive les valeurs numériques des sessions
   PrintFormat("Les heures de la session asiatique sous forme de valeur =%d",AsianSess
   PrintFormat("Les heures de la session européenne sous forme de valeur =%d",Europear
   PrintFormat("Les heures de la session américaine sous forme de valeur =%d",Americar
//--- et affichons maintenant les représentations textuelles des heures de travail des
   Print("Session asiatique ",GetHoursForSession(AsianSession));
   Print("Session européenne ",GetHoursForSession(EuropeanSession));
   Print("Session américaine ",GetHoursForSession(AmericanSession));
//---
  }
//+------------------------------------------------------------------+
//| retourne les heures travaillées de la session sous forme de texte|
//+------------------------------------------------------------------+
string GetHoursForSession(int session)
  {
//--- pour vérifier, utilisez les opérations sur les bits AND et décalage à gauche de
//--- commence par vérifier à partir du plus petit bit
   int bit=1;
   string out="heures de travail : ";
```

© 2000-2024, MetaQuotes Ltd.

```
//--- vérifie les 24 bits en commencant par l'indice 0 et jusqu'au 23
   for(int i=0;i<24;i++)
     {
      //--- récupère l'état de chaque bit sous forme d'un nombre
      bool workinghour=(session&bit)==bit;
      //--- ajoute le numéro de l'heure au message
      if(workinghour )out=out+StringFormat("%d ",i);
      //--- décalage d'un bit vers la gauche pour vérifier la valeur du suivant
      bit<<=1;
     }
//--- chaîne résultat
   return out;
  }
```

Voir aussi

Conversion de Type

© 2000-2024, MetaQuotes Ltd.

Constantes Caractère

Les caractères comme éléments d'une chaîne de caractères en MQL5 sont des indices dans un ensemble de caractères Unicode. Ce sont des valeurs hexadécimales qui peuvent être converties en entiers, et qui peuvent être manipulées par des opérations telles que l'addition et la soustraction.

Tout caractère entre guillemets ou code ASCII hexadécimal d'un caractère comme '\x10' est une constante de caractère et est de type ushort. Par exemple, '0' est la valeur numérique 30, qui correspond à l'indice de zéro dans la table des caractères.

Exemple :

```
void OnStart()
  {
//--- définit les constantes des caractères
   int symbol_0='0';
   int symbol_9=symbol_0+9; // récupère le symbole '9'
//--- affiche les valeurs des constantes
   printf("Sous forme décimale : symbol_0 = %d,  symbol_9 = %d",symbol_0,symbol_9);
   printf("Sous forme hexadécimale : symbol_0 = 0x%x,  symbol_9 = 0x%x",symbol_0,symbo
//--- entre les constantes dans une chaîne
   string test="";
   StringSetCharacter(test,0,symbol_0);
   StringSetCharacter(test,1,symbol_9);
//--- voici à quoi elles ressemblent dans une chaîne de caractères
   Print(test);
  }
```

Le backslash est un caractère de contrôle pour le compilateur lors de l'utilisation de chaînes constantes et de caractères constants dans le texte source d'un programme. Certains symboles, par exemple le guillemet simple ('), le guillemet double ("), le backslash (\) et les caractères de contrôle peuvent être représentés comme une combinaison de symboles commencant avec un backslash (\), selon le tableau ci-dessous :

Nom du caractère	Code mnémonique ou image	Valeur en MQL5	Valeur numérique
nouvelle ligne (line feed)	LF	'\n'	10
tabulation horizontale	HT	'\t'	9
retour chariot	CR	'\r'	13
backslash	\	'\\'	92
guillemet simple	'	'\''	39
double guillemet	"	'\"'	34
code hexadecimal	hhhh	'\xhhhh'	de 1 à 4 caractères hexadécimaux

© 2000-2024, MetaQuotes Ltd.

Nom du caractère	Code mnémonique ou image	Valeur en MQL5	Valeur numérique
code décimal	d	'\d'	nombre décimal de 0 à 65535

Si un backslash est suivi d'un caractère autre que ceux décrits ci-dessus, le résultat est indéfini.

Exemple

```
void OnStart()
  {
//--- déclare les constantes de caractères
   int a='A';
   int b='$';
   int c='©';       // code 0xA9
   int d='\xAE';    // code du symbole ®
//--- affiche les constantes
   Print(a,b,c,d);
//--- ajoute un caractère à la chaîne de caractères
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- remplace un caractère dans une chaîne
   StringSetCharacter(test,0,b);
   Print(test);
//--- remplace un caractère dans une chaîne
   StringSetCharacter(test,0,c);
   Print(test);
//--- remplace un caractère dans une chaîne
   StringSetCharacter(test,0,d);
   Print(test);
//--- représente les caractères sous forme de nombre
   int a1=65;
   int b1=36;
   int c1=169;
   int d1=174;
//--- ajoute un caractère à la chaîne de caractères
   StringSetCharacter(test,1,a1);
   Print(test);
//--- ajoute un caractère à la chaîne de caractères
   StringSetCharacter(test,1,b1);
   Print(test);
//--- ajoute un caractère à la chaîne de caractères
   StringSetCharacter(test,1,c1);
   Print(test);
//--- ajoute un caractère à la chaîne de caractères
   StringSetCharacter(test,1,d1);
   Print(test);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
```

Comme il a été mentionné ci-dessus, la valeur d'une constante de caractère (ou variable) est un indice dans la table des caractères. L'indice étant un entier, il peut être écrit de différentes façons.

```
void OnStart()
  {
//---
   int a=0xAE;      // le code de ® correspond au littéral '\xAE'
   int b=0x24;      // le code de $ correspond au littéral '\x24'
   int c=0xA9;      // le code de © correspond au littéral '\xA9'
   int d=0x263A;    // le code de ☺ correspond au littéral '\x263A'
//--- affiche les valeurs
   Print(a,b,c,d);
//--- ajoute un caractère à la chaîne de caractères
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- remplace un caractère dans une chaîne
   StringSetCharacter(test,0,b);
   Print(test);
//--- remplace un caractère dans une chaîne
   StringSetCharacter(test,0,c);
   Print(test);
//--- remplace un caractère dans une chaîne
   StringSetCharacter(test,0,d);
   Print(test);
//--- codes des cartes
   int a1=0x2660;
   int b1=0x2661;
   int c1=0x2662;
   int d1=0x2663;
//--- ajoute le caractère pique
   StringSetCharacter(test,1,a1);
   Print(test);
//--- ajoute le caractère coeur
   StringSetCharacter(test,2,b1);
   Print(test);
//--- ajoute le caractère carreau
   StringSetCharacter(test,3,c1);
   Print(test);
//--- ajoute le caractère trèfle
   StringSetCharacter(test,4,d1);
   Print(test);
//--- Exemple de littéraux de caractères dans une chaîne
   test="Reine\x2660As\x2662";
   printf("%s",test);
  }
```

© 2000-2024, MetaQuotes Ltd.

La représentation interne d'un littéral de caractère est de type ushort. Les constantes de caractères peuvent accepter des valeurs de 0 à 65535.

Voir aussi

StringSetCharacter(), StringGetCharacter(), ShortToString(), ShortArrayToString(), StringToShortArray()

© 2000-2024, MetaQuotes Ltd.

Type Datetime

Le type datetime est prévu pour stocker la date et l'heure sous la forme du nombre de secondes passées depuis le 1er janvier 1970. Ce type occupe 8 octets en mémoire.

Les constantes de date et heure peuvent être représentées comme une chaîne littérale, qui consiste en 6 parties donnant la valeur numérique de l'année, du mois, du jour (ou jour, mois, année), des heures, des minutes et des secondes. La constante est entourée de guillemets simples et commence avec le caractère D.

Les valeurs s'étalent du 1er janvier 1970 au 31 décembre 3000. La date (année, mois, jour) ou l'heure (heures, minutes, secondes) ou les deux, peuvent être manquants.

Avec la spécification littérale d'une date, il est souhaitable de spécifier l'année, le mois et le jour. Autrement, le compilateur retourne un warning à propos d'une entrée incomplète.

Exemples :

```
datetime NY=D'2015.01.01 00:00';      // Heure de début de l'année 2015
datetime d1=D'1980.07.19 12:30:27';   // Année Mois Jour Heures Minutes Secondes
datetime d2=D'19.07.1980 12:30:27';   // Egal à D'1980.07.19 12:30:27';
datetime d3=D'19.07.1980 12';         // Egal à D'1980.07.19 12:00:00'
datetime d4=D'01.01.2004';            // Egal à D'01.01.2004 00:00:00'
datetime compilation_date=__DATE__;              // Date de compilation
datetime compilation_date_time=__DATETIME__;     // Date et heure de compilation
datetime compilation_time=__DATETIME__-__DATE__;// Heure de compilation
//--- Exemples de déclarations avec lesquels un warning sera généré par le compilateu
datetime warning1=D'12:30:27';        // Egal à D'[date de compilation] 12:30:27'
datetime warning2=D'';                // Egal à __DATETIME__
```

Voir aussi

Structure du Type Date, Date et Heure, TimeToString, StringToTime

© 2000-2024, MetaQuotes Ltd.

Type Color

Le type color est prévu pour stocker les informations de couleur et occupe 4 octets en mémoire. Le premier octet est ignoré, les 3 octets suivants contiennent les composantes RGB.

Les constantes de couleur peuvent être représentées de 3 façons : littéralement, par des entiers ou par un nom (seulement pour les couleurs Web nommées).

La représentation littérale consiste en 3 parties représentant les valeurs numériques des 3 principales composantes : rouge, vert et bleu. La constante commence avec C et est entourée de guillemets simples. Les valeurs numériques de la composante d'une couleur sont dans l'intervalle de 0 à 255.

La représentation par valeur entière est faite sous la forme d'un nombre décimal ou hexadécimal. Un nombre hexadécimal ressemble à 0x00BBGGRR, où RR est la valeur du rouge, GG - celle du vert et BB - celle du bleu. Les constantes décimales ne sont pas directement reflétés en RGB. Elles représentent la valeur décimale de la représentation hexadécimale.

Des couleurs spécifiques correspondent aux couleurs Web.

Exemples :

```
//--- Litteraux
C'128,128,128'    // Gris
C'0x00,0x00,0xFF' // Bleu
// noms des couleurs
clrRed            // Rouge
clrYellow         // Jaune
clrBlack          // Noir
//--- Représentations intégrales
0xFFFFFF          // Blanc
16777215          // Blanc
0x008000          // Vert
32768             // Vert
```

Voir aussi

Couleurs Web, ColorToString, StringToColor, Conversion de Type

© 2000-2024, MetaQuotes Ltd.

Type Bool

Le type bool est prévu pour stocker les valeurs logiques de true ou false, leur représentation numérique est 1 ou 0, respectivement.

Exemples :

```
bool a = true;
bool b = false;
bool c = 1;
```

La représentation interne est un nombre entier de 1 octet. Il est à noter que dans les expressions logiques, vous pouvez utiliser des types entiers ou réels, ou des expressions de ces types - le compilateur ne génèrera aucune erreur. Dans ce cas, la valeur zéro sera interprétée comme fausse (false), et toutes les autres valeurs comme vraie (true).

Exemples :

```
   int i=5;
   double d=-2.5;
   if(i) Print("i = ",i," et vaut true");
   else Print("i = ",i," et vaut false");

   if(d) Print("d = ",d," et a la valeur true");
   else Print("d = ",d," et a la valeur false");

   i=0;
   if(i) Print("i = ",i," et a la valeur true");
   else Print("i = ",i," et a la valeur false");

   d=0.0;
   if(d) Print("d = ",d," et a la valeur true");
   else Print("d = ",d," et a la valeur false");

//--- Résultats d'exécution
//   i= 5 et a la valeur true
//   d= -2.5 et a la valeur true
//   i= 0 et a la valeur false
//   d= 0 et a la valeur false
```

Voir aussi

Opérations Booléennes, Règles de Précédence

© 2000-2024, MetaQuotes Ltd.

Enumérations

Les données du type enum appartiennent à un ensemble limité de données. Définition du type énumération :

```
enum nom du type énumérable
  {
  liste de valeurs
  };
```

La liste des valeurs est une liste d'identifiants de constantes nommées séparées par des virgules.

Exemple :

```
enum mois  // énumération de constantes nommées
   {
   Janvier,
   Février,
   Mars,
   Avril,
   Mai,
   Juin,
   Juillet,
   Août,
   Septembre,
   Octobre,
   Novembre,
   Décembre
   };
```

Une fois que l'énumération est déclarée, un nouveau type de données de 4 octets et valué avec un entier, est disponible. La déclaration d'un nouveau type de données permet au compilateur de contrôler de façon stricte les types des paramètres passés, car l'énumération introduit de nouvelles constantes nommmées. Dans l'exemple ci-dessus, la constante nommée Janvier a la valeur 0, février - 1, décembre - 11.

Règle : si une valeur n'est pas assignée à une constante nommée qui est membre d'une énumération, sa nouvelle valeur sera formée automatiquement. Si c'est le premier membre de l'énumération, la valeur 0 lui est assignée. Pour tous les membres suivants, les valeurs seront calculées sur la base de la valeur des membres précédents en ajoutant 1.

Exemple :

```
enum intervalles // Enumération de constantes nommées
   {
    month=1,      // Intervalle d'un mois
    two_months,   // Deux mois
    quarter,      // Trois mois - trimestre
    halfyear=6,   // Moitié d'une année
    year=12,      // Année - 12 mois
   };
```

© 2000-2024, MetaQuotes Ltd.

Notes

- Contrairement au C++, la taille de la représentation interne du type énuméré en MQL5 est toujours égale à 4 octets. C'est à dire que sizeof(months) retourne la valeur 4.
- Contrairement au C++, une énumération anonyme ne peut pas être déclarée en MQL5. C'est à dire qu'un nom unique doit toujours être spécifié après le mot-clé enum.

Voir aussi

Conversion de Type

© 2000-2024, MetaQuotes Ltd.

Types Réels (double, float)

Les types réels (ou types à virgule flottante) représentent les valeurs ayant une partie fractionnelle. Dans le langage MQL5, il existe deux types de chiffres à virgule flottante. La méthode de représentation des nombres réels dans la mémoire de l'ordinateur est définie par le standard IEEE 754 et est dépendant de la plateforme, du système d'exploitation et du langage de programmation.

Type	Taille en octets	Valeur Positive Minimale	Valeur Maximum	Analogie en C++
float	4	1.175494351e-38	3.402823466e+38	float
double	8	2.2250738585072014e-308	1.7976931348623158e+308	double

Le nom double signifie que la précision de ces nombres est le double de la précision des nombres de type float. Dans la plupart des cas, le type double est le plus pratique. Dans de nombreux cas, la précision limitée des nombres float n'est pas suffisante. La raison de l'utilisation du type float est l'économie de mémoire (c'est important pour les gros tableaux de nombres réels).

Les constantes à virgule flottante consistent en une partie entière, un point (.) et une partie fractionnelle. Les parties entière et fractionnelle sont des suites de chiffres décimaux.

Exemples :

```
double a=12.111;
double b=-956.1007;
float  c =0.0001;
float  d =16;
```

Il existe une façon scientifique d'écrire des constantes réelles, souvent cette méthode est plus compacte.

Exemple :

```
double c1=1.12123515e-25;
double c2=0.000000000000000000000000112123515; // 24 zéros après la virgule

Print("1. c1 =",DoubleToString(c1,16));
// Résultat : 1. c1 = 0.0000000000000000

Print("2. c1 =",DoubleToString(c1,-16));
// Résultat : 2. c1 = 1.1212351499999999e-025

Print("3. c2 =",DoubleToString(c2,-16));
// Résultat : 3. c2 = 1.1212351499999999e-025
```

Il faut se souvenir que les nombres réels sont stockés en mémoire avec une précision limitée dans le système binaire, alors que généralement la notation décimale est utilisée. C'est pourquoi de nombreux nombres qui sont représentés précisemment dans le système décimal ne peuvent être écrit qu'avec une fraction infinie dans le système binaire.

© 2000-2024, MetaQuotes Ltd.

Par exemple, les nombres 0.3 et 0.7 sont représentés dans l'ordinateur comme des fractions infinies, tandhis que 0.25 est stocké de façon exacte car il représente la puissance de 2.

A cet égard, il est fortement recommandé de ne pas comparer deux nombres réels pour l'égalité, car les comparer n'est pas la bonne façon.

Exemple :

```
void OnStart()
  {
//---
   double three=3.0;
   double x,y,z;
   x=1/three;
   y=4/three;
   z=5/three;
   if(x+y==z) Print("1/3 + 4/3 == 5/3");
   else Print("1/3 + 4/3 != 5/3");
// Résultat : 1/3 + 4/3 != 5/3
  }
```

Si vous devez comparer l'égalité de deux nombres réels, vous pouvez le faire de deux façons différentes. La première est de comparer la différence entre deux nombres avec une petite valeur spécifiant la précision de la comparaison.

Exemple :

```
bool EqualDoubles(double d1,double d2,double epsilon)
  {
   if(epsilon<0) epsilon=-epsilon;
//---
   if(d1-d2>epsilon) return false;
   if(d1-d2<-epsilon) return false;
//---
   return true;
  }
void OnStart()
  {
   double d_val=0.7;
   float  f_val=0.7;
   if(EqualDoubles(d_val,f_val,0.000000000000001)) Print(d_val," est égal à ",f_val);
   else Print("Différents : d_val = ",DoubleToString(d_val,16),
              "  f_val = ",DoubleToString(f_val,16));
// Résultat : Différents : d_val= 0.7000000000000000   f_val= 0.6999999880790710
  }
```

Notez que la valeur d'epsilon dans l'exemple ci-dessus ne peut pas être inférieur à la constante prédéfinie DBL_EPSILON. La valeur de cette constante est 2.2204460492503131e-016. La constante correspondant au type float est FLT_EPSILON = 1.192092896e-07. La signification de ces valeurs est la suivante : c'est la plus petite valeur qui satisfait la condition 1.0 + DBL_EPSILON ! = 1.0 (pour les nombres de type float 1.0 + FLT_EPSILON ! = 1.0).

© 2000-2024, MetaQuotes Ltd.

La deuxième façon utilise la comparaison de la différence normalisée de deux nombres réels avec zéro. Il est inutile de comparer la différence des nombres normalisés avec zéro, car toute opération mathématique avec des nombres normalisés donne un résultat non normalisé.

Exemple :

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0) return(true);
   else return(false);
  }
void OnStart()
  {
   double d_val=0.3;
   float  f_val=0.3;
   if(CompareDoubles(d_val,f_val)) Print(d_val," égal ",f_val);
   else Print("Différents : d_val = ",DoubleToString(d_val,16),
              "  f_val = ",DoubleToString(f_val,16));
// Résultat : Différents : d_val= 0.3000000000000000   f_val= 0.3000000119209290
  }
```

Certaines opérations du coprocesseur mathématique peuvent générer des nombres r&els invalides, qui ne peuvent pas être utilisés dans les opérations mathématiques et dans les comparaisons, car le résultat d'opération avec des nombres réels est indéfini. Par exemple, lors du calcul de l'arc-sinus de 2, le résultat est l'infini négatif.

Exemple :

```
   double abnormal = MathArcsin(2.0);
   Print("MathArcsin(2.0) =",abnormal);
// Résultat :  MathArcsin(2.0) = -1.#IND
```

En plus de l'infini négatif, il y a également l'infini positif et NaN (NotANumber, pas un nombre). Pour savoir si un nombre est invalide, vous pouvez appeler la fonction MathIsValidNumber(). Selon le standard IEEE, ils ont une représentation machine spéciale. Par exemple, l'infini positif pour le type double est représenté par 0x7FF0 0000 0000 0000.

Exemples :

```
struct str1
  {
   double d;
  };
struct str2
  {
   long l;
  };

//--- Début
   str1 s1;
   str2 s2;
```

© 2000-2024, MetaQuotes Ltd.

```
//---
   s1.d=MathArcsin(2.0);        // Calcule le nombre invalide -1.#IND
   s2=s1;
   printf("1.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFFF000000000000;     // nombre invalide -1.#QNAN
   s1=s2;
   printf("2.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF7000000000000;     // plus grand SNaN (NotANumber)
   s1=s2;
   printf("3.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF8000000000000;     // plus petit QNaN
   s1=s2;
   printf("4.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FFF000000000000;     // plus grand QNaN
   s1=s2;
   printf("5.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF0000000000000;     // Infini positif 1.#INF et plus petit SNaN
   s1=s2;
   printf("6.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFF0000000000000;     // Infini négatif -1.#INF
   s1=s2;
   printf("7.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x8000000000000000;     // Zéro négatif -0.0
   s1=s2;
   printf("8.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FE0000000000000;     // 0.5
   s1=s2;
   printf("9.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FF0000000000000;     // 1.0
   s1=s2;
   printf("10.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FEFFFFFFFFFFFFF;     // Plus grand nombre normalisé (MAX_DBL)
   s1=s2;
   printf("11.  %.16e %I64X",s1.d,s2.l);
//---
   s2.l=0x0010000000000000;     // Plus petit nombre positif normalisé (MIN_DBL)
   s1=s2;
   printf("12.  %.16e %.16I64X",s1.d,s2.l);
//---
```

© 2000-2024, MetaQuotes Ltd.

```
   s1.d=0.7;                    // Montre que le nombre 0.7 est une fraction infinie
   s2=s1;
   printf("13.  %.16e %.16I64X",s1.d,s2.l);
/*
1.  -1.#IND00 FFF8000000000000
2.  -1.#QNAN0 FFFF000000000000
3.   1.#SNAN0 7FF7000000000000
4.   1.#QNAN0 7FF8000000000000
5.   1.#QNAN0 7FFF000000000000
6.   1.#INF00 7FF0000000000000
7.  -1.#INF00 FFF0000000000000
8.  -0.000000 8000000000000000
9.   0.500000 3FE0000000000000
10.  1.000000 3FF0000000000000
11.  1.7976931348623157e+308 7FEFFFFFFFFFFFFF
12.  2.2250738585072014e-308 0010000000000000
13.  6.9999999999999996e-001 3FE6666666666666
*/
```

Voir aussi

DoubleToString, NormalizeDouble, Constantes de Type Numérique

© 2000-2024, MetaQuotes Ltd.

Nombre complexe (complex)

Le type complex intégré est une structure à deux champs double :

```
struct complex
  {
   double              real;   // Partie réelle
   double              imag;   // Partie imaginaire
  };
```

Le type "complex" peut être passé par valeur comme paramètre pour les fonctions MQL5 (contrairement aux structures ordinaires, qui ne sont passées que par référence). Pour les fonctions importées à partir de DLL, le type "complex" doit être passé uniquement par référence.

Le suffixe 'i' est utilisé pour décrire des constantes complexes :

```
complex square(complex c)
  {
   return(c*c);
  }
void OnStart()
  {
   Print(square(1+2i));  // Une constante est passée en paramètre
  }
// "(-3,4)" will be output, which is a string representation of the complex number
```

Seules les opérations simples sont actuellement disponibles pour les nombres complexes : =, +, -, *, /, +=, -=, *=, /=, ==, !=.

La prise en charge de fonctions mathématiques supplémentaires sera ajoutée ultérieurement, permettant le calcul de la valeur absolue, du sinus, du cosinus et autres.

© 2000-2024, MetaQuotes Ltd.

Type String

Le type string est utilisé pour stocker des chaînes de caractères. Une chaîne de caractères est une suite de caractères au format Unicode avec un zéro terminal à la fin. Une chaîne de caractères constante peut être affectée à une variable de chaîne de caractères. Une chaîne de caractères constante est une suite de caractères Unicode entourée de double guillemets : "Ceci est une chaîne de caractères constante".

Si vous avez besoin d'inclure un double guillemets dans une chaîne de caractères, il faut utiliser le caractère backslash (\) juste avant le caractère. Toute constante de caractère spéciale peut être écrite dans une chaîne si un backslash (\) est placé juste devant.

Exemples :

```
string svar="Ceci est une chaîne de caractères";
string svar2=StringSubstr(svar,0,4);
Print("Symbole copyright\t\x00A9");
FileWrite(handle,"Cette chaîne de caractères contient un symbole nouvelle ligne \n");
string MT5path="C:\\Program Files\\MetaTrader 5";
```

Pour rendre le code source lisible, les longues chaînes de caractères constantes peuvent être découpées en plusieurs parties sans opération d'addition. Pendant la compilation, ces parties seront assemblées en une seule longue chaîne de caractères :

```
//--- Déclare une longue chaîne de caractères
   string HTML_head="<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\""
                    " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n"
                    "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n"
                    "<head>\n"
                    "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ut
                    "<title>Rapport des Opérations de Trading</title>\n"
                    "</head>";
//--- Ecrit les chaînes de caractères dans le journal
   Print(HTML_head);
  }
```

Voir aussi

Fonctions de Conversion, Fonctions de Chaînes de Caractères, FileOpen, FileReadString, FileWriteString

© 2000-2024, MetaQuotes Ltd.

Structures, Classes et Interfaces

Structures

Une structure est un ensemble d'éléments de n'importe quel type (sauf le type void). Une structure regroupe donc des données logiques de différents types entre elles.

Déclaration de la Structure

Le type de données structure est défini par la description suivante :

```
struct structure_name
  {
   elements_description
  };
```

Le nom de la structure ne peut pas être utilisé comme un identifiant (nom d'une variable ou d'une fonction). Il est à noter que dans une structure MQL5, les éléments se suivent les uns après les autres, sans alignement. En C++, cet ordre est réalisé par le compilateur avec l'instruction suivante :

```
#pragma pack(1)
```

Si vous souhaitez réaliser un autre alignement dans la structure, utilisez des membres auxiliaires appelés "fillers" et ayant la bonne taille.

Exemple :

```
struct trade_settings
  {
   uchar  slippage;     // valeur de la taille admissible du slippage 1 octet
   char   reserved1;    // saute 1 octet
   short  reserved2;    // saute 2 octets
   int    reserved4;    // 4 autres octets sont sautés pour assurer l'alignement sur 8
   double take;         // valeur du prix de fixation du profit
   double stop;         // valeur du prix du stop de protection
  };
```

Cette description des structures alignées n'est nécessaire que pour le transfert à des fonctions de dll importées.

Attention : cette exemple illustre des données conçue de façon incorrecte. Il serait mieux de déclarer en premier les données les plus grandes : le *take* et *stop* de type double, et ensuite de déclarer le membre du *slippage* de type uchar. Dans ce cas, la représentation interne des données sera toujours la même, indépendamment de la valeur spécifiée dans l'instruction *#pragma pack()*.

Si une structure contient des variables de type string et/ou un tableau dynamique, le compilateur crée automatiquement un constructeur implicite. Ce constructeur réinitialise tous les membres de la structure de type string et initialise correctement les tableaux dynamiques.

Structures Simples

© 2000-2024, MetaQuotes Ltd.

Les structures qui ne contiennent pas de chaînes de caractères, d'objets de classe, de pointeurs ou de tableaux dynamiques sont appelées des structures simples. Les variables de structures simples, de même que leurs tableaux, peuvent être passées en paramètres aux fonctions de DLL importées.

La copie de structures simples n'est autorisée que dans deux cas :

- Si les objets appartiennent à la même structure
- si les objets sont connectés par un héritage, c'est à dire qu'une structure est dérivée d'une autre.

Pour fournir un exemple, développons la structure CustomMqlTick avec un contenu identique à la structure de base MqlTick. Le compilateur ne permet pas de copier la valeur de l'objet MqlTick dans un objet de type CustomMqlTick. La conversion directe vers le type nécessaire cause également une erreur de compilation :

```
      //--- la copie de structures simples de différents types est interdite
      my_tick1=last_tick;               // le compilateur retourne une erreur ici

      //--- la conversion de structures de différents types les unes vers les autres e
      my_tick1=(CustomMqlTick)last_tick;// le compilateur retourne une erreur ici
```

Il ne reste donc qu'une seule option - la copie de la valeur des éléments de la structure une par une. Il est toujours autorisé de copier les valeurs du même type de CustomMqlTick.

```
      CustomMqlTick my_tick1,my_tick2;
      //--- il est autorisé de copier les objets du même type de CustomMqlTick de la f
      my_tick2=my_tick1;

      //--- crée un tableau à partir des objets de la structure simple CustomMqlTick
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
```

La fonction ArrayPrint() est appelée pour vérification en affichant la *valeur du tableau arr[] dans le journal*.

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- développe la structure similaire à la structure intégrée MqlTick
   struct CustomMqlTick
     {
      datetime          time;          // Dernière heure de mise à jour du prix
      double            bid;           // Prix Bid courant
      double            ask;           // Prix Ask courant
      double            last;          // Prix courant de la dernière transaction (Las
      ulong             volume;        // Volume du prix Last courant
      long              time_msc;      // Dernière mise à jour du prix Last en millise
      uint              flags;         // Flags du tick
     };
   //--- récupère la valeur du dernier tick
```

© 2000-2024, MetaQuotes Ltd.

```
   MqlTick last_tick;
   CustomMqlTick my_tick1,my_tick2;
//--- tentative de copier les données de MqlTick vers CustomMqlTick
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      //--- la copie de structures simples sans aucun rapport est interdite
      //1. my_tick1=last_tick;               // le compilateur retourne une erreur ici

      //--- il est également interdit de convertir entre elles des structures sans rap
      //2. my_tick1=(CustomMqlTick)last_tick;// le compilateur retourne une erreur ici

      //--- il faut donc copier les membres de la structure un par un
      my_tick1.time=last_tick.time;
      my_tick1.bid=last_tick.bid;
      my_tick1.ask=last_tick.ask;
      my_tick1.volume=last_tick.volume;
      my_tick1.time_msc=last_tick.time_msc;
      my_tick1.flags=last_tick.flags;

      //--- il est autorisé de copier les objets du même type de CustomMqlTick de la f
      my_tick2=my_tick1;

      //--- crée un tableau à partir des objets de la structure simple CustomMqlTick
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
      ArrayPrint(arr);
//--- exemple d'affichage des valeurs du tableau contenant les objets de type CustomMq
      /*
                       [time]    [bid]    [ask]    [last] [volume]     [time_msc] [flags]
      [0] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157       2
      [1] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157       2
      */
     }
   else
      Print("Echec de SymbolInfoTick(), erreur = ",GetLastError());
  }
```

Le deuxième exemple montre la copie de structures simples par héritage. Supposons que nous avons la structure de base Animal, à partir de laquelle les deux structures Cat et Dog sont dérivées. Nous pouvons copier les objets Animal et Cat, ainsi que les objets Animal et Dog les uns vers les autres, mais nous ne pouvons pas copier Cat et Dog l'un vers l'autre, bien qu'ils soient tous les deux descendants de la structure Animal.

```
//--- structure pour décrire les chiens
struct Dog: Animal
  {
   bool              hunting;       // race de chien de chasse
  };
```

© 2000-2024, MetaQuotes Ltd.

```
//--- structure pour décrire les chats
struct Cat: Animal
  {
   bool              home;          // race domestique
  };
//--- crée les objets des structures filles
   Dog dog;
   Cat cat;
//--- peut être copié de l'ancêtre vers le descendant (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // les chiens ne peuvent pas nager
//--- vous ne pouvez pas copier les objets des structures filles (Dog != Cat)
   cat=dog;        // le compilateur retourne une erreur
```

Code d'exemple complet :

```
//--- structure de base pour décrire les animaux
struct Animal
  {
   int               head;          // nombre de têtes
   int               legs;          // nombre de jambes
   int               wings;         // nombre d'ailes
   bool              tail;          // queue
   bool              fly;           // voler
   bool              swim;          // nager
   bool              run;           // courir
  };
//--- structure pour décrire les chiens
struct Dog: Animal
  {
   bool              hunting;       // race de chien de chasse
  };
//--- structure pour décrire les chats
struct Cat: Animal
  {
   bool              home;          // race domestique
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée et décrit un objet du type de base Animal
   Animal some_animal;
   some_animal.head=1;
   some_animal.legs=4;
   some_animal.wings=0;
   some_animal.tail=true;
   some_animal.fly=false;
```

© 2000-2024, MetaQuotes Ltd.

```
   some_animal.swim=false;
   some_animal.run=true;
//--- crée les objets des types enfants
   Dog dog;
   Cat cat;
//--- peut être copié de l'ancêtre vers le descendant (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // les chiens peuvent nager
//--- vous ne pouvez pas copier les objets des structures filles (Dog != Cat)
   //cat=dog;        // le compilateur retourne une erreur ici
//--- il faut donc copier les éléments un par un
   cat.head=dog.head;
   cat.legs=dog.legs;
   cat.wings=dog.wings;
   cat.tail=dog.tail;
   cat.fly=dog.fly;
   cat.swim=false;   // les chats ne peuvent pas nager
//--- il est possible de copier les valeurs du descendant vers l'ancêtre
   Animal elephant;
   elephant=cat;
   elephant.run=false;// les éléphants ne peuvent pas courir
   elephant.swim=true;// les éléphants peuvent nager
//--- crée un tableau
   Animal animals[4];
   animals[0]=some_animal;
   animals[1]=dog;
   animals[2]=cat;
   animals[3]=elephant;
//--- affichage
   ArrayPrint(animals);
//--- résultat de l'exécution
/*
      [tête] [jambes] [ailes] [queue] [vole] [nage] [court]
   [0]      1       4       0   true false  false  true
   [1]      1       4       0   true false   true  true
   [2]      1       4       0   true false  false false
   [3]      1       4       0   true false   true false
*/
  }
```

Une autre façon de copier les types simples est d'utiliser une union. Les objets des structures doivent être membres de la même union - voir l'exemple dans la section union.

Accès aux Membres d'une Structure

Le nom d'une structure devient un nouveau type de données, vous pouvez donc déclarer des variables de ce type. La structure ne peut être déclarée qu'une seule fois dans un projet. Les membres de la structures sont disponibles en utilisant l'opérateur point (.).

© 2000-2024, MetaQuotes Ltd.

Exemple :

```
struct trade_settings
  {
   double take;          // valeur du prix de fixation du profit
   double stop;          // valeur du prix du stop de protection
   uchar  slippage;      // valeur du slippage autorisé
  };
//--- crée et initialise une variable de type trade_settings
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

'pack' pour aligner les champs d'une structure et d'une classe

L'attribut spécial pack permet de définir l'alignement de champs d'une structure ou d'une classe.

```
 pack([n])
```

où n est l'une des valeurs suivantes : 1, 2, 4, 8 ou 16. Il peut être omis.

Exemple :

```
   struct pack(sizeof(long)) MyStruct
     {
      // les membres seront alignés sur 8 octets
     };
ou
   struct MyStruct pack(sizeof(long))
     {
      // les membres seront alignés sur 8 octets
     };
```

'pack(1)' est appliqué par défaut pour les structures. Cela signifie que les membres de la structure sont situés les uns parès les autres en mémoire, et que la taille de la structure est égale à la somme de la taille de ses membres.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- structure simple sans alignement
   struct Simple_Structure
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- déclare l'instance d'une structure simple
```

© 2000-2024, MetaQuotes Ltd.

```
   Simple_Structure s;
//--- affiche la taille de chaque membre de la structure
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- s'assure que la taille de la structure POD est égale à la somme de la taille de
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  Résultat :
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=15
*/
  }
```

Il peut être nécessaire d'aligner les champs de la structure lors de l'echange de données avec les bibliothèques tierces (*.DLL) où ce type d'alignement est appliqué.

Utilisons quelques exemples pour montrer comment l'alignement fonctionne. Nous appliquerons une structure constituée de 4 membres sans alignement.

```
//--- structure simple sans alignement
   struct Simple_Structure pack() // aucune taille n'est spécifiée, l'alignement sur 1
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
//--- -- déclare l'instance d'une structure simple
   Simple_Structure s;
```

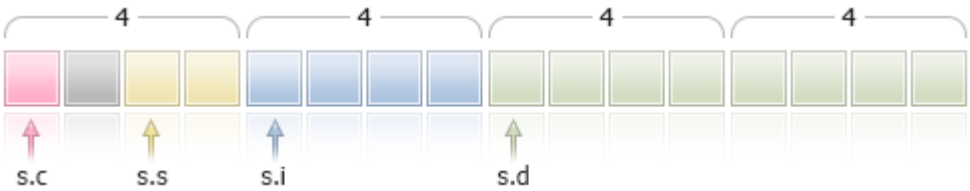
Les champs de la structure doivent être situé en mémoire les uns après les autres selon l'ordre de déclaration et la taille du type. La taille de la structure est 15, et le décalage des champs de la structure dans les tableaux est non défini.

Déclarons maintenant la même structure avec un alignement sur 4 octets, et exécutons le code.

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
```

```
//--- structure simple avec un alignement sur 4 octets
   struct Simple_Structure pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- déclare l'instance d'une structure simple
   Simple_Structure s;
//--- affiche la taille de chaque membre de la structure
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- assurons nous que la taille de la structure POD est maintenant égale à la somme
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  Résultat :
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=16 // la taille de la structure a changé
*/
  }
```

La taille de la structure a changé afin que tous les membres de 4 octets ou plus ont un décalage d'un multiple de 4 depuis le début de la structure. Les petits membres sont alignés sur la limite de leur propre taille (par exemple, 2 pour 'short'). Cela ressemble à ça (l'octet ajouté est affiché en gris).



Dans ce cas, 1 octet est ajouté après le membre **s.c**, afin que le champ **s.s** (sizeof(short)==2) ait une limite de 2 octets (alignement pour le type 'short').

Le décalage depuis le début de la structure dans le tableau est également aligné sur 4 octets, c'est à dire que les adresses des éléments a[0], a[1] et a[n] sont des multiples de 4 octets pour Simple_Structure arr[].

Considérons 2 structures supplémentaires constituée de types similaires avec un alignement de 4 octets mais avec un ordre différent des membres. Dans la première structure, les membres sont ordonnés par la taille croissante du type.

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- structure simple alignée sur 4 octets
   struct CharShortInt pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(double)=4
     };
//--- -- déclare l'instance d'une structure simple
   CharShortInt ch_sh_in;
//--- affiche la taille de chaque membre de la structure
   Print("sizeof(ch_sh_in.c)=",sizeof(ch_sh_in.c));
   Print("sizeof(ch_sh_in.s)=",sizeof(ch_sh_in.s));
   Print("sizeof(ch_sh_in.i)=",sizeof(ch_sh_in.i));

//--- s'assure que la taille de la structure POD est égale à la somme de la taille de
   Print("sizeof(CharShortInt)=",sizeof(CharShortInt));
/*
  Résultat :
   sizeof(ch_sh_in.c)=1
   sizeof(ch_sh_in.s)=2
   sizeof(ch_sh_in.i)=4
   sizeof(CharShortInt)=8
*/
  }
```

Comme nous pouvons le voir, la taille de la structure est 8 et est faite de 2 blocs de 4 octets. Le premier bloc contient les champs avec les types 'char' et 'short', et le second contient le champ avec le type 'int'.

Transformons maintenant la 1ère structure dans la 2ème, qui ne diffère que par l'ordre des champs, en déplacant le membre de type 'short' à la fin.

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- structure simple alignée sur 4 octets
   struct CharIntShort pack(4)
     {
      char              c; // sizeof(char)=1
      int               i; // sizeof(double)=4
      short             s; // sizeof(short)=2
```

© 2000-2024, MetaQuotes Ltd.

```
     };
//--- -- déclare l'instance d'une structure simple
   CharIntShort ch_in_sh;
//--- affiche la taille de chaque membre de la structure
   Print("sizeof(ch_in_sh.c)=",sizeof(ch_in_sh.c));
   Print("sizeof(ch_in_sh.i)=",sizeof(ch_in_sh.i));
   Print("sizeof(ch_in_sh.s)=",sizeof(ch_in_sh.s));
//--- s'assure que la taille de la structure POD est égale à la somme de la taille de
   Print("sizeof(CharIntShort)=",sizeof(CharIntShort));
/*
  Résultat :
   sizeof(ch_in_sh.c)=1
   sizeof(ch_in_sh.i)=4
   sizeof(ch_in_sh.s)=2
   sizeof(CharIntShort)=12
*/
  }
```

Bien que le contenu de la structure n'ait pas changé, modifier l'ordre des membres a augmenté sa taille.

Il faut également tenir compte de l'alignement lors de l'héritage. Démontrons-le avec une structure simple Parent ayant un seul membre de type 'char'. La taille de la structure sans alignement est 1.

```
   struct Parent
     {
      char              c;    // sizeof(char)=1
     };
```

Créons la classe fille Children comprenant un membre de type 'short' (sizeof(short)=2).

```
   struct Children pack(2) : Parent
     {
      short             s;   // sizeof(short)=2
     };
```

Avec un alignement sur 2 octets, la taille de la structure est égale à 4, bien que la taille de ses membres soit 3. Dans cet exemple, 2 octets sont alloués sur la classe Parent, afin que l'accès au champ 'short' de la classe fille soit aligné sur 2 octets.

La connaissance de l'allocation mémoire pour les membres de la structure est nécessaire si une application MQL5 interagit avec des données tierces en écrivant/lisant les fichiers ou les flux.

Le répertoire MQL5\Include\WinAPI de la Bibliothèque Standard contient les fonctions pour travailler avec les fonctions WinAPI. Ces fonctions appliquent les structures avec un alignement spécifié pour les cas où il est requis de travailler avec WinAPI.

© 2000-2024, MetaQuotes Ltd.

offsetof est une commande spéciale directement reliée à l'attribut pack. Cela nous permet d'obtenir le décalage d'un membre à partir du début de la structure.

```
//--- déclare la variable de type Children
   Children child;
//--- détecte les décalages depuis le début de la structure
   Print("offsetof(Children,c)=",offsetof(Children,c));
   Print("offsetof(Children,s)=",offsetof(Children,s));
/*
  Résultat :
   offsetof(Children,c)=0
   offsetof(Children,s)=2
*/
```

Spécificateur 'final'

L'utilisation du spécificateur 'final' dans la déclaration d'une structure interdit tout héritage ultérieur de cette structure. Si une structure ne requiert aucune modification ultérieurement, ou que les modifications ne sont pas autorisées pour des raisons de sécurité, déclarez cette structure avec le spécificateur 'final'. De plus, tous les membres de la structure seront également considérés final de façon implicite.

```
struct settings final
  {
  //--- Corps de la structure
  };

struct trade_settings : public settings
  {
  //--- Corps de la structure
  };
```

Si vous tentez d'hériter d'une structure ayant le modificateur 'final' comme montré dans l'exemple ci-dessus, le compilateur retournera une erreur :

```
cannot inherit from 'settings' as it has been declared as 'final'
see declaration of 'settings'
```

Classes

Les classes sont différentes des structures :

- le mot-clé class est utilisé dans la déclaration ;
- par défaut, tous les membres de la classe sont considérés comme privés, sauf mention contraire. Les données membres de la structure sont considérées comme publiques par défaut, sauf mention contraire ;
- les instances de classe ont toujours une table des fonctions virtuelles, même s'il n'y a aucune fonction virtuelle déclarée dans la classe. Les structures ne peuvent pas avoir de fonctions virtuelles ;
- l'opérateur new peut être appliqués aux classes ; cet opérateur ne peut pas être appliqué aux structures ;

© 2000-2024, MetaQuotes Ltd.

- les classes ne peuvent dériver que d'autres classes, les structures ne peuvent dériver que d'autres structures.

Les classes et les structures peuvent avoir un constructeur et un destructeur explicite. Si votre constructeur est défini explicitement, l'initialisation de l'instance d'une sructure ou d'une classe utilisant la séquence d'initialisation n'est pas possible.

Exemple :

```
struct trade_settings
  {
   double take;          // valeur du prix de fixation du profit
   double stop;          // valeur du prix du stop de protection
   uchar  slippage;      // valeur du slippage autorisé
   //--- Constructeur
          trade_settings() { take=0.0; stop=0.0; slippage=5; }
   //--- Destructeur
         ~trade_settings() { Print("C'est la fin"); }
  };
//--- Le compilateur générera un message d'erreur comme quoi l'initialisation n'est pa
trade_settings my_set={0.0,0.0,5};
```

Constructeurs et Destructeurs

Un constructeur est une fonction spéciale appelée automatiquement lors de la création de l'instance d'une structure ou d'une classe et est habituellement utilisé pour initialiser les membres de la classe. Nous parlerons par la suite de classes, mais les mêmes notions s'appliquent aux structures, sauf mention contraire. Le nom d'un constructeur doit correspondre au nom de la classe. Le constructeur n'a pas de type de retour (vous pouvez spécifier le type void).

Les membres de classe définis - chaîne de caractères, tableaux dynamiques et objets nécessitant une initialisation - seront dans tous les cas initialisés, indépendamment du fait qu'il y ait un constructeur ou pas.

Chaque classe peuvent avoir plusieurs constructeurs, différant par le nombre de paramètres et la liste d'initialisation. Un constructeur nécessitant de spécifier des paramètres est appelé un constructeur paramétré.

Un constructeur sans paramètre est appelé **un constructeur par défaut**. Si aucun constructeur n'est déclaré dans une classe, le compilateur crée un constructeur par défaut pendant la compilation.

```
//+------------------------------------------------------------------+
//| Classe pour travailler avec une date                             |
//+------------------------------------------------------------------+
class MyDateClass
  {
private:
   int               m_year;          // Année
   int               m_month;         // Mois
   int               m_day;           // Jour du mois
   int               m_hour;          // Heure
   int               m_minute;        // Minutes
```

© 2000-2024, MetaQuotes Ltd.

```
   int                m_second;         // Secondes
public:
   //--- Constructeur par défaut
                      MyDateClass(void);
   //--- Constructeur paramétré
                      MyDateClass(int h,int m,int s);
  };
```

Un constructeur peut être déclaré dans la description de la classe et son corps peut ensuite être défini. Par exemple, 2 constructeurs de MyDateClass peuvent être définis de la façon suivante :

```
//+------------------------------------------------------------------+
//| Constructeur par défaut                                          |
//+------------------------------------------------------------------+
MyDateClass::MyDateClass(void)
  {
//---
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=mdt.hour;
   m_minute=mdt.min;
   m_second=mdt.sec;
   Print(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Constructeur paramétré                                           |
//+------------------------------------------------------------------+
MyDateClass::MyDateClass(int h,int m,int s)
  {
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=h;
   m_minute=m;
   m_second=s;
   Print(__FUNCTION__);
  }
```

Dans le constructeur par défaut, tous les membres de la classes sont remplis avec la fonction TimeCurrent(). Dans le constructeur paramétré, seules les heures sont remplies. Les autres membres de la classe (m_year, m_month et m_day) seront initialisés automatiquement avec la date courante.

© 2000-2024, MetaQuotes Ltd.

Le constructeur par défaut a une fonction spéciale lors de l'initialisation d'un tableau d'instances de cette classe. Le constructeur dont tous les paramètres ont des valeurs par défaut, **n'est pas** un constructeur par défaut. Voici un exemple :

```
//+------------------------------------------------------------------+
//| Une classe avec un constructeur par défault                      |
//+------------------------------------------------------------------+
class CFoo
  {
   datetime          m_call_time;     // Heure du dernier appel à l'objet
public:
   //--- Constructeur avec un paramètre ayant une valeur par défaut n'est pas un const
                     CFoo(const datetime t=0){m_call_time=t;};
   //--- Constructeur par copie
                     CFoo(const CFoo &foo){m_call_time=foo.m_call_time;};

   string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
// CFoo foo; // Cette variante ne peut pas être utilisée - un constructeur par défaut
//--- Options possibles pour créer un objet CFoo
   CFoo foo1(TimeCurrent());      // Un appel explicite à un constructeur paramétré
   CFoo foo2();                   // Un appel explicite à un constructeur paramétré ave
   CFoo foo3=D'2009.09.09';       // Un appel implicite à un constructeur paramétré
   CFoo foo40(foo1);              // Un appel explicite à un constructeur par copie
   CFoo foo41=foo1;               // Un appel implicite à un constructeur par copie
   CFoo foo5;                     // Un appel explicite à un constructeur par défaut (s
                                  // alors un constructeur paramétré avec une valeur pa
//--- Options possibles pour reçevoir des pointeurs de CFoo
   CFoo *pfoo6=new CFoo();        // Création dynamique d'un objet et affectation d'un
   CFoo *pfoo7=new CFoo(TimeCurrent());// Autre option de création d'un objet dynamiqu
   CFoo *pfoo8=GetPointer(foo1); // Maintenant pfoo8 pointe sur l'objet foo1
   CFoo *pfoo9=pfoo7;             // pfoo9 et pfoo7 pointent sur le même objet
   // CFoo foo_array[3];          // Cette option ne peut pas être utilisée - un constr
//--- Affiche la valeur de m_call_time
   Print("foo1.m_call_time=",foo1.ToString());
   Print("foo2.m_call_time=",foo2.ToString());
   Print("foo3.m_call_time=",foo3.ToString());
   Print("foo4.m_call_time=",foo4.ToString());
   Print("foo5.m_call_time=",foo5.ToString());
   Print("pfoo6.m_call_time=",pfoo6.ToString());
   Print("pfoo7.m_call_time=",pfoo7.ToString());
   Print("pfoo8.m_call_time=",pfoo8.ToString());
   Print("pfoo9.m_call_time=",pfoo9.ToString());
//--- Supprime les tableaux créés dynamiquement
```

© 2000-2024, MetaQuotes Ltd.

```
   delete pfoo6;
   delete pfoo7;
   //delete pfoo8;  // Vous n'avez pas à supprimer pfoo8 explicitement, puisque il po
   //delete pfoo9;  // Vous n'avez pas à supprimer pfoo8 explicitement puisqu'il oiune
  }
```

Si vous décommentez ces lignes

```
  //CFoo foo_array[3];     // Cette variante ne peut pas être utilisée - il n'y a aucu
```

ou

```
  //CFoo foo_dyn_array[];  // Cette variante ne peut pas être utilisée - il n'y a aucu
```

alors le compilateur retournera une erreur pour chaque ligne : "default constructor is not defined".

Si une classe a un constructeur défini, le constructeur par défaut n'est pas généré par le compilateur. Cela signifie que si un constructeur paramétré est déclaré dans une classe, et qu'aucun constructeur n'est déclaré, vous ne pouvez pas déclarer de tableaux d'instances de cette classe. Le compilateur retournera une erreur pour ce script :

```
//+------------------------------------------------------------------+
//| Une classe sans constructeur par défaut                          |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
                     CFoo(string name) { m_name=name;}
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Génère l'erreur "default constructor is not defined" pendant la compilation
   CFoo badFoo[5];
  }
```

Dans cet exemple, la classe CFoo a un constructeur paramétré - dans ce cas, le compilateur ne crée pas automatiquement un constructeur par défaut pendant la compilation. En même temps, lorsque vous déclarez un tableau d'instances, il est supposé que tous les objets sont créés et initialisés automatiquement. Pendant l'initialisation automatique d'un objet, il est nécessaire d'appeler un constructeur par défaut, mais puisque le constructeur par défaut n'est pas déclaré explicitement et n'est pas généré automatiquement par le compilateur non plus, il est impossible de créer cet objet. Pour cette raison, le compilateur génère une erreur au moment de la compilation.

Il existe une syntaxe spéciale pour initialiser un objet en utilisant un constructeur. Les initialiseurs (constructions spéciales pour l'initialisation) des membres d'une struct ou d'une classe peuvent être spécifiées dans la liste d'initialisation.

© 2000-2024, MetaQuotes Ltd.

Une liste d'initialisation est une liste d'initialiseurs séparés par des virgules, venant après le symbole ':', après la liste des paramètres d'un constructeur et avant le corps (se place avant une accolade ouvrante). Il y a plusieurs contraintes :

- Les listes d'initialisation ne peuvent être utilisées que dans les constructeurs;
- Les membres parents ne peuvent pas être initialisés dans la liste d'initialisation ;
- La liste d'initialisation doit être suivie par la définition (implémentation) de la function.

Voici un exemple de plusieurs constructeurs initialisant les membres de la classe.

```
//+------------------------------------------------------------------+
//| Une classe stockant le nom d'un personnage                       |
//+------------------------------------------------------------------+
class CPerson
  {
   string            m_first_name;     // Premier nom
   string            m_second_name;    // Deuxième nom
public:
   //--- Un constructeur par défaut vide
                     CPerson() {Print(__FUNCTION__);};
   //--- Un constructeur paramétré
                     CPerson(string full_name);
   //--- Un constructeur avec une liste d'initialisation
                     CPerson(string surname,string name): m_second_name(surname), m_f
   void PrintName(){PrintFormat("Name=%s Surname=%s",m_first_name,m_second_name);};
  };
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
CPerson::CPerson(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>=0)
     {
      m_first_name=StringSubstr(full_name,0,pos);
      m_second_name=StringSubstr(full_name,pos+1);
     }
  }
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Génère l'erreur "default constructor is not defined" (aucun constructeur par dé
   CPerson people[5];
   CPerson Tom="Tom Sawyer";                        // Tom Sawyer
   CPerson Huck("Huckleberry","Finn");              // Huckleberry Finn
   CPerson *Pooh = new CPerson("Winnie","Pooh");  // Winnie l'Ourson
   //--- Valeurs de sortie
```

© 2000-2024, MetaQuotes Ltd.

```
   Tom.PrintName();
   Huck.PrintName();
   Pooh.PrintName();

   //--- Supprime l'objet créé dynamiquement
   delete Pooh;
  }
```

Dans ce cas, la classe CPerson a 3 constructeurs :

1. Un constructeur par défaut explicite, permettant de créer un tableau d'instances de cette classe ;
2. Un constructeur a 1 paramètre, prenant en argument le nom complet et le découpe en nom et prénom en fonction de l'espace trouvé ;
3. Un constructeur a 2 paramètres contenant une liste d'initialisation. Initialiseurs - m_second_name(surname) et m_first_name(name).

Noter que l'initialisation utilisant une liste a remplacé une affectation. Les membres individuels doit être initialisés comme :

```
 class_member (une liste d'expressions)
```

Dans la liste d'initialisation, les membres peuvent être dans n'importe quel ordre, mais tous les membres de la classe seront initialisés suivant leur ordre dans la déclaration. Cela signifie que dans le troisième constructeur, le membre m_first_name sera d'abord initialisé, puisqu'il est déclaré en premier, et seulement après, m_second_name sera initialisé. Ceci doit être pris en compte dans les cas om l'initialisation de certains membres de la classe dépendent de valeurs d'autres membres de la classe.

Si aucun constructeur par défaut n'est déclaré dans la classe de base, et qu'au moins un constructeur avec des paramètres est déclaré, vous devriez toujours appeler l'un des constructeurs de la classe de base dans la liste d'initialisation. Il se place entre les virgules comme des membres ordinaires de la liste et sera appelé en premier pendant l'initialisation de l'objet, indépendamment de son placement dans la liste d'initialisation.

```
//+------------------------------------------------------------------+
//| Classe de base                                                   |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
   //--- Un constructeur avec une liste d'initialisation
                     CFoo(string name) : m_name(name) { Print(m_name);}
  };
//+------------------------------------------------------------------+
//| Classe dérivée de CFoo                                           |
//+------------------------------------------------------------------+
class CBar : CFoo
  {
   CFoo              m_member;      // Le membre de classe est un objet de la classe p
public:
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- Un constructeur par défaut dans la liste d'initialisation appelle le construc
                    CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
   CBar bar;
  }
```

Dans cet exemple, lors de la création de l'objet bar, le constructeur par défaut CBar() sera appelé, dans lequel le constructeur du parent CFoo est d'abord appelé, et ensuite le constructeur du membre de classe m_member.

Un destructeur est une fonction spéciale, appelée automatiquement lorsqu'un objet est détruit. Le nom du destructeur est le nom de la classe avec un tilde (~). Les chaînes de caractères, les tableaux dynamiques et les objets, nécessitant une désinitialisation, seront désinitialisés quand même, indépendamment de la présence ou non du destructeur. Si un destructeur existe, ces actions seront effectuées après l'appel au destructeur.

Les destructeurs sont toujours virtuels, qu'ils soient déclarés avec le mot-clé virtual ou pas.

Définir des Méthodes de Classe

Les méthodes-fonctions de classe peuvent être définies dans ou en dehors de la déclaration de la classe. Si la méthode est définie dans la classe, alors son corps vient juste après la déclaration de la méthode.

Exemple :

```
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
```

© 2000-2024, MetaQuotes Ltd.

```
   int               GetType()                   { return(m_type);         }
   void              Left()                      { m_xpos-=SHAPE_SIZE;     }
   void              Right()                     { m_xpos+=SHAPE_SIZE;     }
   void              Rotate()                    { m_prev_turn=m_turn; if(++m_turn>3) r
   virtual void      Draw()                      { return;                 }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

Les fonctions de SetRightBorder(int border) à Draw() sont déclarées et définies directement dans la classe CTetrisShape.

Le constructeur CTetrisShape() et les méthodes CheckDown(int& pad_array[]), CheckLeft(int& side_row[]) et CheckRight(int& side_row[]) sont seulement déclarées dans la classe, mais pas encore définies. Les définitions de ces fonctions seront placés plus loin dans le code. Pour définir la méthode en dehors de la classe, l'opérateur de résolution de portée est utilisé, le nom de la classe est utilisé pour la portée.

Exemple :

```
//+------------------------------------------------------------------+
//| Constructeur de la classe de base                                |
//+------------------------------------------------------------------+
void CTetrisShape::CTetrisShape()
  {
   m_type=0;
   m_ypos=0;
   m_xpos=0;
   m_xsize=SHAPE_SIZE;
   m_ysize=SHAPE_SIZE;
   m_prev_turn=0;
   m_turn=0;
   m_right_border=0;
  }
//+------------------------------------------------------------------+
//| Vérifie si la barre et le cube peuvent descendre                 |
//+------------------------------------------------------------------+
bool CTetrisShape::CheckDown(int& pad_array[])
  {
   int i,xsize=m_xsize/SHAPE_SIZE;
//---
   for(i=0; i<xsize; i++)
     {
      if(m_ypos+m_ysize>=pad_array[i]) return(false);
     }
//---
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

Modificateurs d'Accès Public, Protected et Private

Lors du développement d'une nouvelle classe, il est recommandé de restreindre l'accès aux membres depuis l'extérieur. Pour cela, il faut utiliser les mots-clés private ou protected. Dans ce cas, les données cachées ne peuvent être utilisées que depuis les méthodes-fonctions de la même classe. Si le mot-clé *protected* est utilisé, les données cachées peut être accédées depuis les méthodes des classes héritant de cette classe. La même méthode peut être utilisée pour restreindre l'accès aux fonctions-méthodes de la classe.

Si vous avez besoin d'ouvrir complètement l'accès aux membres et/ou méthodes d'une classe, utilisez le mot-clé public.

Exemple :

```
class CTetrisField
  {
private:
   int              m_score;                          // Score
   int              m_ypos;                           // Position actuelle des figu
   int              m_field[FIELD_HEIGHT][FIELD_WIDTH]; // Matrice de la zone de jeu
   int              m_rows[FIELD_HEIGHT];             // Numération des lignes de l
   int              m_last_row;                       // Dernière ligne vide
   CTetrisShape    *m_shape;                          // Figure du Tétris
   bool             m_bover;                          // Perdu
public:
   void             CTetrisField() { m_shape=NULL; m_bover=false; }
   void             Init();
   void             Deinit();
   void             Down();
   void             Left();
   void             Right();
   void             Rotate();
   void             Drop();
private:
   void             NewShape();
   void             CheckAndDeleteRows();
   void             LabelOver();
  };
```

Tous les membres de classe et les méthodes déclarées après le spécificateur **public:** (et avant le spécificateur d'accès suivant) sont disponibles dans toute occurrence de l'objet dans le programme. Dans cet exemple, ce sont les membres suivants : fonctions CTetrisField(), Init(), Deinit(), Down(), Left(), Right(), Rotate() et Drop().

Tous les membres déclarés après le spécificateur d'accès **private:** (et avant le spécificateur d'accès suivant) ne sont disponibles qu'aux fonctions membres de cette classe. Les spécificateurs d'accès aux élements se terminent toujours avec le symbole ':' et peuvent apparaître plusieurs fois dans la définition de la classe.

Tous les membres de classe déclarés après le spécificateur d'accès **protected:** (et jusqu'au prochain spécificateur d'accès) ne sont disponibles que pour les fonctions membres de cette classe et des

© 2000-2024, MetaQuotes Ltd.

classes descendantes. En essayant de faire référence aux membres comportant les spécificateurs private et protected de l'extérieur, nous obtenons une erreur de compilation. Exemple :

```
class A
  {
protected:
   //--- l'opérateur de copie n'est diposnible qu'à l'intérieur de la classe A et de s
   void operator=(const A &)
     {
     }
  };
class B
  {
   //-- objet de classe A déclaré
   A                 a;
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- déclare 2 variables de type B
   B b1, b2;
   //--- tente de copier l'un des objets dans l'autre
   b2=b1;
  }
```

Lors de la compilation du code, le message d'erreur est retourné — tentative d'appeler l'opérateur de copie distant :

```
attempting to reference deleted function 'void B::operator=(const B&)'   trash3.mq5
```

La deuxième chaîne de caractères en dessous est une description plus détaillée — l'opérateur de copie dans la classe B a été supprimé explicitement, puisque l'opérateur indisponible de copie de la classe A est appelé :

```
   function 'void B::operator=(const B&)' was implicitly deleted because it invokes in
```

L'accès aux membres de la classe de base peut être redéfini pendant l'héritage dans les classes dérivées.

Spécificateur 'delete'

Le spécificateur delete marque les fonctions membres de la classe qui ne peuvent pas être utilisées. Cela signifie que si le programme se réfère à une telle fonction explicitement ou implicitement, l'erreur de compilation est générée. Par exemple, ce spécificateur vous permet de rendre les méthodes parentes indisponibles dans une classe fille. Le même résultat peut être obtenu si nous déclarons la fonction dans la zone privée de la classe parente (déclarations dans la section private). Ici, l'utilisation de delete rend le code plus lisible et plus facile à utiliser au niveau des descendants.

```
class A
  {
```

© 2000-2024, MetaQuotes Ltd.

```
public:
                        A(void) {value=5;};
   double               GetValue(void) {return(value);}
private:
   double               value;
  };
class B: public A
  {
   double               GetValue(void)=delete;
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare une variable de type A
   A a;
   Print("a.GetValue()=", a.GetValue());
//--- essaye d'obtenir la valeur de la variable de type B
   B b;
   Print("b.GetValue()=", b.GetValue()); // le compilateur affiche une erreur sur cett
  }
```

Le message du compilateur :

```
attempting to reference deleted function 'double B::GetValue()'
   function 'double B::GetValue()' was explicitly deleted here
```

Le spécificateur 'delete' permet de désactiver la conversion automatique ou le constructeur par copie, qui autrement aurait été caché dans la section private également. Exemple :

```
class A
  {
public:
   void                 SetValue(double v) {value=v;}
   //--- désactive l'appel de type int
   void                 SetValue(int) = delete;
   //--- désactive l'opérateur de copie
   void                 operator=(const A&) = delete;
private:
   double               value;
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare 2 variables de type A
   A a1, a2;
   a1.SetValue(3);       // erreur !
```

© 2000-2024, MetaQuotes Ltd.

```
   a1.SetValue(3.14);    // OK
   a2=a1;                // erreur !
  }
```

Pendant la compilation, nous obtenons les messages d'erreur :

```
attempting to reference deleted function 'void A::SetValue(int)'
   function 'void A::SetValue(int)' was explicitly deleted here
attempting to reference deleted function 'void A::operator=(const A&)'
   function 'void A::operator=(const A&)' was explicitly deleted here
```

Spécificateur 'final'

L'utilisation du modificateur 'final' dans la déclaration de la classe interdit tout héritage ultérieur de cette classe. Si l'interface de la classe ne nécessite aucune autre modification, ou si les modifications ne sont pas autorisées pour des raisons de sécurité, declarez cette classe avec le spécificateur 'final'. De plus, tous les membres de la classe seront également considérés 'final' de façon implicite.

```
class CFoo final
  {
  //--- Corps de la classe
  };

class CBar : public CFoo
  {
  //--- Corps de la classe
  };
```

Si vous tentez d'hériter d'une classe ayant le modificateur 'final' comme montré dans l'exemple ci-dessus, le compilateur retournera une erreur :

```
cannot inherit from 'CFoo' as it has been declared as 'final'
see declaration of 'CFoo'
```

Unions (union)

L'union est un type spécial de données, consistant en plusieurs variables partageant le même segment de mémoire. L'union fournit donc la possibilité d'interpréter la même séquence de bits de 2 (ou plus) façons différentes. La déclaration d'une union est similaire à la déclaration d'une structure et commence avec le mot-clé union.

```
union LongDouble
{
  long   long_value;
  double double_value;
};
```

Contrairement aux structures, tous les membres de l'nunion appartiennent au même segment de mémoire. Dans cet exemple, l'union de LongDouble est déclarée avec des valeurs de type long et double partageant la même zone de mémoire. Veuillez noter qu'il est impossible que l'union stocke une valeur entière de type *long* et une valeur réelle de type *double* simultanément (contrairement à une structure), puisque les variables long_value et double_value se chevauchent (en mémoire). D'un autre côté, un programme MQL5 est capable de traiter des données contenant dans l'union un entier (long)

© 2000-2024, MetaQuotes Ltd.

ou un réel (double) n'importe quand. L'union permet donc d'avoir 2 (ou plus) options pour représenter la même séquence de données.

Pendant la déclaration de l'union, le compilateur alloue automatiquement suffisamment d'espace mémoire pour stocker le plus grand type (en terme de volume) dans la variable. La même syntaxe est utilisée pour accéder à l'élément de l'union comme pour les structures - l'opérateur point.

```
union LongDouble
{
  long   long_value;
  double double_value;
};
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   LongDouble lb;
//--- récupère et affiche le nombre invalide -nan(ind)
   lb.double_value=MathArcsin(2.0);
   printf("1.  double=%f                integer=%I64X",lb.double_value,lb.long_value);
//--- plus grande valeur normalisée (DBL_MAX)
   lb.long_value=0x7FEFFFFFFFFFFFFF;
   printf("2.  double=%.16e  integer=%I64X",lb.double_value,lb.long_value);
//--- plus petite valeur positive normalisée (DBL_MIN)
   lb.long_value=0x0010000000000000;
   printf("3.  double=%.16e  integer=%.16I64X",lb.double_value,lb.long_value);
  }
/*  Résultat de l'exécution
    1.  double=-nan(ind)                integer=FFF8000000000000
    2.  double=1.7976931348623157e+308  integer=7FEFFFFFFFFFFFFF
    3.  double=2.2250738585072014e-308  integer=0010000000000000
*/
```

Puisque les unions permettent au programme d'interpréter les mêmes données en mémoire de différentes façons, elles sont souvent utilisées lorsqu'une conversion de type est requise.

Les unions peuvent être impliquées dans l'héritage, et elles ne peuvent pas non plus avoir de membres statiques en raison de leur nature. Dans tous les autres aspects, l'*union* se comporte comme une structure, avec tous ses membres sans offset. Les types suivants ne peuvent pas être membre d'une union :

- tableaux dynamiques
- chaînes de caractères
- pointeurs d'objets et de fonctions
- objets de classe
- objets structure ayant des constructeurs et des destructeurs
- objets structure ayant des membres des points 1-5

© 2000-2024, MetaQuotes Ltd.

Comme les classes, l'union peut avoir des constructeurs et des destructeurs, ainsi que des méthodes. Par défaut, les membres de l'union ont un type d'accès public. Pour pouvoir créer des éléments privés, utilisez le mot-clé private. Toutes ces possibilités sont affichées dans l'exemple illustrant comment convertir une couleur du type color en ARGB comme le fait la fonction ColorToARGB().

```
//+------------------------------------------------------------------+
//| Union pour la conversion d'une couleur (BGR) en ARGB             |
//+------------------------------------------------------------------+
union ARGB
  {
   uchar             argb[4];
   color             clr;
   //--- constructeurs
                     ARGB(color col,uchar a=0){Color(col,a);};
                    ~ARGB(){};
   //--- méthodes publiques
public:
   uchar   Alpha(){return(argb[3]);};
   void    Alpha(const uchar alpha){argb[3]=alpha;};
   color   Color(){ return(color(clr));};
   //--- méthodes privées
private:
   //+----------------------------------------------------------------+
   //| place la valeur du canal alpha et la couleur                   |
   //+----------------------------------------------------------------+
   void    Color(color col,uchar alpha)
     {
      //--- définit la couleur du membre clr
      clr=col;
      //--- définit la valeur du composant Alpha - niveau d'opacité
      argb[3]=alpha;
      //--- intervertit les octets des composantes R et B (Rouge et Bleu)
      uchar t=argb[0];argb[0]=argb[2];argb[2]=t;
     };
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 0x55 signifie 55/255=21.6 % (0% - transparence totale)
   uchar alpha=0x55;
//--- le type color est représenté par 0x00BBGGRR
   color test_color=clrDarkOrange;
//--- les valeurs des octets de l'union ARGB sont acceptées ici
   uchar argb[];
   PrintFormat("0x%.8X - voici comment le type 'color' est représenté pour %s, BGR=(%s
               test_color,ColorToString(test_color,true),ColorToString(test_color));
//--- le type ARGB est représenté comme 0x00RRGGBB, les composants RR et BB sont inter
```

© 2000-2024, MetaQuotes Ltd.

```
   ARGB argb_color(test_color);
//--- copie le tableau d'octets
   ArrayCopy(argb,argb_color.argb);
//--- voici comment il ressemble dans la représentation ARGB
   PrintFormat("0x%.8X - représentation ARGB avec le canal alpha=0x%.2x, ARGB=(%d,%d,%
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- ajoute le niveau d'opacité
   argb_color.Alpha(alpha);
//--- essaye de définir ARGB comme type 'color'
   Print("ARGB comme color=(",argb_color.clr,")  canal alpha=",argb_color.Alpha());
//--- copie le tableau d'octets
   ArrayCopy(argb,argb_color.argb);
//--- voici comment il ressemble dans la représentation ARGB
   PrintFormat("0x%.8X - représentation ARGB avec le canal alpha=0x%.2x, ARGB=(%d,%d,%
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- vérifie avec le résultat de la fonction ColorToARGB()
   PrintFormat("0x%.8X - résultat de ColorToARGB(%s,0x%.2x)",ColorToARGB(test_color,al
               ColorToString(test_color,true),alpha);
  }
/*  Résultat de l'exécution
   0x00008CFF - voici comment le type color est représenté pour clrDarkOrange, BGR=(25
  0x00FF8C00 - représentation ARGB avec le canal alpha=0x00, ARGB=(0,255,140,0)
   ARGB comme color=(0,140,255)  canal alpha=85
   0x55FF8C00 - représentation ARGB avec le canal alpha=0x55, ARGB=(85,255,140,0)
   0x55FF8C00 - résultat de ColorToARGB(clrDarkOrange,0x55)
*/
```

Interfaces

Une interface permet de déterminer une fonctionnalité spécifique, qu'une classe peut ensuite implémenter. En fait, une interface est une classe qui ne peut pas contenir de membre, et n'a pas de constructeur et de destructeur. Toutes les méthodes déclarées dans une interface sont purement virtuelle, même sans définition explicite.

Une interface est définie avec le mot-clé "interface". Exemple :

```
//--- Interface de base décrivant les animaux
interface IAnimal
  {
//--- Les méthodes de l'interface ont l'accès public par défaut
   void Sound();  // Le son produit par l'animal
  };
//+------------------------------------------------------------------+
//|  La classe CCat hérite de l'interface IAnimal                    |
//+------------------------------------------------------------------+
class CCat : public IAnimal
  {
public:
                      CCat() { Print("Le chat est né"); }
```

© 2000-2024, MetaQuotes Ltd.

```
                    ~CCat() { Print("Le chat est mort");  }
   //--- Implémentation de la méthode Sound de l'interface IAnimal
   void Sound(){ Print("meou"); }
  };
//+------------------------------------------------------------------+
//|  La classe CDog hérite de l'interface IAnimal                    |
//+------------------------------------------------------------------+
class CDog : public IAnimal
  {
public:
                     CDog() { Print("Le chien est né"); }
                    ~CDog() { Print("le chien est mort");  }
   //--- Implémentation de la méthode Sound de l'interface IAnimal
   void Sound(){ Print("guaf"); }
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Un tableau de pointeurs vers des objets de type IAnimal
   IAnimal *animals[2];
//--- Création de classes filles de IAnimal et sauvegarde des pointeurs sur ces objets
   animals[0]=new CCat;
   animals[1]=new CDog;
//--- Appel de la méthode Sound() de l'interface de base IAnimal pour chaque enfant
   for(int i=0;i<ArraySize(animals);++i)
      animals[i].Sound();
//--- Suppression des objets
   for(int i=0;i<ArraySize(animals);++i)
      delete animals[i];
//--- Résultat de l'exécution
/*
  Le chat est né
  Le chien est né
   meou
   guaf
  Le chat est mort
  Le chien est mort
*/
  }
```

Comme pour les classes abstraites, un objet interface ne peut pas être créé sans héritage. Une interface ne peut hériter que d'autres interfaces et peut être le parent de classes. Une interface est toujours visible publiquement.

Une interface ne peut pas être déclarée dans la déclaration d'une classe ou d'une structure, mais un pointeur sur l'interface peut être sauvé dans une variable du type void *. De façon générale, un pointeur sur un objet de n'importe quelle classe peut être sauvé dans une variable du type void *. Pour

© 2000-2024, MetaQuotes Ltd.

convertir un pointeur void * vers un pointeur sur un objet d'une classe particulière, utilisez l'opérateur dynamic_cast. Si la conversion n'est pas possible, le résultat de l'opération dynamic_cast sera NULL.

Voir aussi

Programmation Orientée Objet

© 2000-2024, MetaQuotes Ltd.

Objet Tableau Dynamique

Tableaux Dynamiques

Des tableaux de 4 dimensions maximum peuvent être déclarés. Lors de la déclaration d'un tableau dynamique (un tableau avec une valeur non spécifiée pour les 2 premières paires de crochets), le compilateur crée automatiquement une variable de la structure ci-dessus (un objet tableau dynamique) et fournit le code pour une initialisation correcte.

Les tableaux dynamiques sont libérés automatiquement lors de la sortie du bloc dans lesquels ils sont déclarés.

Exemple :

```
double matrix[][10][20]; // tableau dynamique à 3 dimensions
ArrayResize(matrix,5);   // Définit la taille de la première dimension
```

Tableaux Statiques

Lorsque toutes les dimensions significatives du tableau sont explicitement spécifiées, le compilateur pré-alloue la taille suffisante de mémoire. Ce type de tableau est appelé statique. Néanmoins, le compilateur alloue de la mémoire supplémentaire pour l'objet d'un tableau dynamique, qui est associé avec le buffer statique pré-alloué (partie de la mémoire pour stocker le tableau).

La création d'un objet tableau dynamique est dû au besoin possible de passer ce tableau statique comme paramètre à certaines fonctions.

Exemples :

```
double stat_array[5]; // tableau statique à 1 dimension
some_function(stat_array);
...
bool some_function(double& array[])
  {
   if(ArrayResize(array,100)<0) return(false);
   ...
   return(true);
  }
```

Tableaux dans les Structures

Lorsqu'un tableau statique est déclaré comme membre d'une structure, l'objet du tableau dynamique n'est pas créé. Ceci est fait pour assurer la compatibilité des structures de données utilisées dans les API de Windows.

Cependant, les tableaux statiques qui sont déclarés comme membres de structures peuvent également être passés aux fonctions MQL5. Dans ce cas, lors du passage du paramètre, un objet temporaire de tableau dynamique sera créé. Un tel objet est lié avec le tableau statique - membre de structure.

Voir aussi

Fonctions de Tableaux, Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Création et Suppression des Objets

© 2000-2024, MetaQuotes Ltd.

Matrices et vecteurs

Le type vector est un type de données spécial dans MQL5, qui permet des opérations avec des vecteurs. Un vecteur est un tableau à une dimension de type double. C'est l'un des concepts fondamentaux de l'algèbre linéaire, qui est utilisé dans de nombreux domaines scientifiques, notamment la physique, la géométrie et d'autres. Les vecteurs sont utilisés pour résoudre des systèmes d'équations linéaires, dans les graphiques 3D et dans d'autres domaines d'application. Les vecteurs peuvent être ajoutés et multipliés. La longueur ou la distance entre les vecteurs peut être obtenue grâce à la Norme. En programmation, les vecteurs sont généralement représentés par des tableaux d'éléments homogènes, qui peuvent ne pas avoir d'opérations vectorielles régulières, c'est-à-dire lorsque les tableaux ne peuvent pas être additionnés ou multipliés et qu'ils n'ont pas de norme.

Les vecteurs peuvent être représentés sous forme de vecteurs de ligne et de vecteurs de chaîne lorsque vous travaillez avec des matrices. De plus, les vecteurs en algèbre linéaire utilisent les concepts de covariance et de contravariance. Ces concepts ne font aucune différence lors de l'écriture d'un code MQL5, car seul le programmeur décide de ce qu'est chaque objet de type vectoriel. Par exemple, il peut s'agir d'un vecteur de rotation, de déplacement ou de compression dans les graphiques 3D.

D'une manière générale, du point de vue de l'algèbre linéaire, un nombre est aussi un vecteur, mais dans un espace vectoriel à une dimension. Un vecteur lui-même peut être considéré comme un cas particulier de matrice.

Le type matrix est un autre type de données spécial dans MQL5 pour représenter des matrices. Une matrice est en fait un tableau à deux dimensions de type double. Les vecteurs et des matrices ont été introduits dans MQL5 pour faciliter les opérations avec certains types de jeux de données. Avec eux, les développeurs peuvent bénéficier des possibilités d'algèbre linéaire sous une forme simple et mathématique. Les matrices peuvent être utilisées pour écrire de manière compacte des systèmes d'équations linéaires ou différentielles. Le nombre de lignes de la matrice correspond au nombre d'équations, tandis que le nombre de colonnes est égal au nombre d'inconnues. En conséquence, les systèmes d'équations linéaires peuvent être résolus par des opérations matricielles.

Les types de données suivants existent :

- matrix — une matrice contenant des éléments de type double.
- matrixf — une matrice contenant des éléments de type float.
- matrixc — une matrice contenant des éléments de type complexe.
- vector — un vecteur contenant des éléments de type double.
- vectorf — un vecteur contenant des éléments de type float.
- vectorc — un vecteur contenant des éléments de type complexe.

Les fonctions templatées prennent en charge les notations telles que matrix<double>, matrix<float>, vector<double>, vector<float> au lieu des types correspondants.

Les opérations algébriques suivantes sont définies pour les matrices :

- Ajout de matrices de même taille
- Multiplication de matrices de taille appropriée : le nombre de colonnes de la matrice de gauche doit être égal au nombre de lignes de la matrice de droite
- Multiplication matricielle par un vecteur colonne ; multiplication d'un vecteur ligne par une matrice selon la règle de multiplication matricielle. En ce sens, le vecteur est un cas particulier d'une matrice

© 2000-2024, MetaQuotes Ltd.

- Multiplication matricielle par un nombre, c'est-à-dire par un scalaire

Les mathématiques considèrent de nombreux types de matrices différents. Par exemple, matrice d'identité, matrices symétriques, asymétriques, triangulaires supérieure et inférieure et autres types. Diverses formes Normales jouent un rôle important dans la théorie des matrices. Elles représentent une certaine forme canonique d'une matrice, qui peut être obtenue au moyen de certaines transformations. En pratique, on utilise des formes normales qui ont des propriétés supplémentaires, comme par exemple la stabilité.

L'utilisation de vecteurs et de matrices, ou plutôt de méthodes spéciales des types concernés, permet de créer un code plus simple, plus bref et plus clair, proche de la notation mathématique. Avec ces méthodes, vous pouvez éviter d'avoir à créer des boucles imbriquées ou de vous soucier de l'indexation correcte des tableaux dans les calculs. Par conséquent, l'utilisation de ces méthodes augmente la fiabilité et la vitesse de développement de programmes complexes.

Liste des méthodes matricielles et vectorielles

Les types matrix et vector incluent des méthodes qui correspondent aux méthodes NumPy correspondantes de la bibliothèque. En utilisant ces méthodes, vous pouvez traduire des algorithmes et des codes de Python en MQL5 avec un minimum d'efforts. De nombreuses tâches de traitement de données, d'équations mathématiques, de réseaux de neurones et de tâches d'apprentissage automatique peuvent être résolus à l'aide de méthodes et de bibliothèques Python prêtes à l'emploi.

Méthode Matrice/vecteur	Méthode analogue dans NumPy	Description
void matrix.Eye(const int rows, const int cols, const int ndiag=0)	eye	Construit une matrice avec des 1 sur la diagonale et des 0 ailleurs
void matrix.Identity(const int rows)	identity	Construit une matrice carrée avec des unités sur la diagonale principale
void matrix.Ones(const int rows, const int cols)	ones	Construit une nouvelle matrice de lignes et de colonnes données, remplies de 1
void matrix.Zeros(const int rows, const int cols)	zeros	Construit une nouvelle matrice de lignes et de colonnes données, remplies de 0
void matrix.Full(const int rows, const int cols, const scalar value)	full	Construit une nouvelle matrice de lignes et de colonnes données, remplies d'une valeur scalaire
void matrix.Copy(const matrix a)	copy	Construit une copie de la matrice donnée
void matrix.FromBuffer(const int rows, const int cols, const scalar array[], const int count=-1, const int offset=0)	frombuffer	Construit une matrice créée à partir d'un tableau à 1 dimension

© 2000-2024, MetaQuotes Ltd.

Méthode Matrice/vecteur	Méthode analogue dans NumPy	Description
void matrix.FromFile(const int rows, condt int cols, const int file_handle, const int count=-1, const int offset=0)	fromfile	Construit une matrice à partir des données d'un fichier texte ou binaire
void vector.FromString(const string source, const string sep=" ")	fromstring	Construit un vecteur initialisé à partir de données texte dans une chaîne
void vector.Arange(const scalar start, const scalar stop, const scalar step=1)	arange	Construit des valeurs régulièrement espacées dans un intervalle donné
void matrix.Diag(const vector v, const int ndiag=0)	diag	Extrait une diagonale ou construit une matrice diagonale
void matrix.Tri(const int rows, const int cols, const int ndiag=0)	tri	Construit une matrice avec des 1 au niveau et en dessous de la diagonale donnée et des 0 ailleurs
void matrix.Tril(const int rows, const int cols, const scalar array[], const int ndiag=0)	tril	Renvoie une copie d'une matrice avec des éléments au-dessus de la k-ième diagonale mis à zéro
void matrix.Triu(const int rows, const int cols, const scalar array[], const int ndiag=0)	triu	Renvoie une copie d'une matrice avec les éléments sous la k-ième diagonale mis à zéro
void matrix.Vander(const vector v, const int cols=-1, const bool increasing=false)	vander	Générer une matrice de Vandermonde
vector matrix.Row(const unsigned nrow)		Renvoie un vecteur ligne
vector matrix.Col(const unsigned ncol)		Renvoie un vecteur colonne
unsigned matrix.Rows()		Renvoie le nombre de lignes d'une matrice
unsigned matrix.Cols()		Renvoie le nombre de colonnes d'une matrice
void matrix.Init()		Initialise une matrice
matrix matrix.Transpose()	transpose	Inverse ou permute les axes d'une matrice ; renvoie la matrice modifiée
matrix matrix.Dot(const matrix b)	dot	Produit scalaire de deux matrices
matrix matrix.Inner(const matrix b)	inner	Produit scalaire de deux matrices

© 2000-2024, MetaQuotes Ltd.

Méthode Matrice/vecteur	Méthode analogue dans NumPy	Description
matrix matrix.Outer(const matrix b)	outer	Calcule le produit extérieur de deux matrices
matrix matrix.MatMul(const matrix b)	matmul	Produit matriciel de deux matrices
matrix matrix.MatrixPower(const int power)	matrix_power	Éleve une matrice carrée à la puissance (entière) n
matrix matrix.Kron(const matrix b)	kron	Renvoie le produit de Kronecker de deux matrices
bool matrix.Cholesky(matrix& L)	cholesky	Renvoie la décomposition de Cholesky
bool matrix.QR(matrix& Q, matrix& R)	qr	Calcule la factorisation qr d'une matrice
bool matrix.SVD(matrix& U, matrix& V, vector& singular_values)	svd	Décomposition en valeur singulière
bool matrix.Eig(matrix& eigen_vectors, vector& eigen_values)	eig	Calcule les valeurs propres et les vecteurs propres droits d'une matrice carrée
bool matrix.EigH(matrix& eigen_vectors, vector& eigen_values)	eigh	Renvoie les valeurs propres et vecteurs propres d'une matrice Hermitienne
bool matrix.EigVals(vector& eigen_values)	eigvals	Calcule les valeurs propres d'une matrice générale
bool matrix.EigValsH(vector& eigen_values)	eigvalsh	Calcule les valeurs propres d'une matrice Hermitienne
bool matrix.LU(matrix& L, matrix& U)		Décomposition LU d'une matrice comme le produit d'une matrice triangulaire inférieure et d'une matrice triangulaire supérieure
bool matrix.LUP(matrix& L, matrix& U, matrix& P)		Décomposition LUP avec pivotement partiel, qui fait référence à la décomposition LU avec permutations de lignes uniquement : PA=LU
double matrix.Norm(const norm)	norm	Renvoie la matrice ou le vecteur normé
double matrix.Cond(const norm)	cond	Calcule le nombre de conditions d'une matrice
vector matrix.Spectrum()		Calcule le spectre d'une matrice comme l'ensemble de ses valeurs

© 2000-2024, MetaQuotes Ltd.

Méthode Matrice/vecteur	Méthode analogue dans NumPy	Description
		propres à partir du produit AT*A
double matrix.Det()	det	Calcule le déterminant d'un tableau
int matrix.Rank()	matrix_rank	Retourne le rang matriciel du tableau à l'aide de la méthode Gaussienne
int matrix.SLogDet(int& sign)	slogdet	Calcule le signe et le logarithme du déterminant d'un tableau
double matrix.Trace()	trace	Renvoie la somme le long des diagonales de la matrice
vector matrix.Solve(const vector b)	solve	Résoud une équation matricielle linéaire ou un système d'équations algébriques linéaires
vector matrix.LstSq(const vector b)	lstsq	Renvoie la solution des moindres carrés des équations algébriques linéaires (pour les matrices non carrées ou dégénérées)
matrix matrix.Inv()	inv	Calcule l'inverse (multiplicatif) d'une matrice
matrix matrix.PInv()	pinv	Calcule le pseudo-inverse d'une matrice par la méthode de Moore-Penrose
int matrix.Compare(const matrix matrix_c, const double epsilon) int matrix.Compare(const matrix matrix_c, const int digits) int vector.Compare(const vector vector_c, const double epsilon) int vector.Compare(const vector vector_c, const int digits)		Compare les éléments de deux matrices/vecteurs avec la précision demandée
double matrix.Flat(const ulong index) bool matrix.Flat(const ulong index,const double value)	flat	Permet d'adresser un élément de matrice via un index au lieu de deux
double vector.ArgMax() double matrix.ArgMax() vector matrix.ArgMax(const int axis)	argmax	Renvoie l'indice de la valeur maximale
double vector.ArgMin() double matrix.ArgMin() vector matrix.ArgMin(const int axis)	argmin	Renvoie l'indice de la valeur minimale

© 2000-2024, MetaQuotes Ltd.

Méthode Matrice/vecteur	Méthode analogue dans NumPy	Description
double vector.Max() double matrix.Max() vector matrix.Max(const int axis)	max	Renvoie la valeur maximale dans une matrice/vecteur
double vector.Mean() double matrix.Mean() vector matrix.Mean(const int axis)	mean	Calculer la moyenne arithmétique des valeurs des éléments
double vector.Min() double matrix.Min() vector matrix.Min(const int axis)	min	Renvoie la valeur minimale dans une matrice/vecteur
double vector.Sum() double matrix.Sum() vector matrix.Sum(const int axis)	sum	Renvoie la somme des éléments de matrice/vecteur qui peuvent également être exécutés pour l'axe (les axes) donné(s).
void vector.Clip(const double min_value,const double max_value) void matrix.Clip(const double min_value,const double max_value)	clip	Limite les éléments d'une matrice/d'un vecteur à un intervalle spécifié de valeurs valides
vector vector.CumProd() vector matrix.CumProd() matrix matrix.CumProd(const int axis)	cumprod	Retourne le produit cumulatif des éléments d'une matrice/d'un vecteur, y compris ceux le long de l'axe donné
vector vector.CumSum() vector matrix.CumSum() matrix matrix.CumSum(const int axis)	cumsum	Renvoie la somme cumulée des éléments d'une matrice/d'un vecteur, y compris ceux le long de l'axe donné
double vector.Prod(const double initial=1) double matrix.Prod(const double initial=1) vector matrix.Prod(const int axis,const double initial=1)	prod	Renvoie le produit des éléments d'une matrice/d'un vecteur qui peut également être effectué pour l'axe donné
void matrix.Reshape(const ulong rows, const ulong cols)	reshape	Change la forme d'une matrice sans changer ses données
void matrix.Resize(const ulong rows,const ulong cols)	resize	Renvoie une nouvelle matrice avec une forme et une taille modifiées
bool matrix.SwapRows(const ulong row1, const ulong row2)		Échanger des lignes dans une matrice
bool matrix.SwapCols(const ulong col1, const ulong col2)		Échanger des colonnes dans une matrice

© 2000-2024, MetaQuotes Ltd.

Méthode Matrice/vecteur	Méthode analogue dans NumPy	Description
double vector.Ptp() double matrix.Ptp() vector matrix.Ptp(const int axis)	ptp	Renvoie la plage de valeurs d'une matrice/d'un vecteur ou de l'axe de la matrice donné, équivalent à Max() - Min()
double vector.Percentile(const int percent) double matrix.Percentile(const int percent) vector matrix.Percentile(const int percent,const int axis)	percentile	Renvoie le centile spécifié des valeurs des éléments d'une matrice/d'une vecteur ou des éléments le long de l'axe spécifié. Les valeurs valides du paramètre 'percent' sont comprises dans la plage [0, 100]
double vector.Quantile(const int percent) double matrix.Quantile(const int percent) vector matrix.Quantile(const int percent,const int axis)	quantile	Renvoie le quantile spécifié des valeurs des éléments d'une matrice/d'un vecteur ou des éléments le long de l'axe spécifié. Le paramètre 'percent' prend des valeurs dans la plage [0, 1]
double vector.Median() double matrix.Median() vector matrix.Median(const int axis)	median	Calcule la médiane des éléments d'une matrice/d'un vecteur. La médiane est la valeur médiane qui sépare la moitié la plus élevée des éléments du tableau/vecteur de la moitié la plus basse des éléments.
double vector.Average() double matrix.Average() vector matrix.Average(const int axis)	average	Calculer la moyenne arithmétique des valeurs d'une matrice/vecteur. La somme des poids du dénominateur ne peut pas être égale à 0, mais certains poids peuvent être 0
double vector.Std() double matrix.Std() vector matrix.Std(const int axis)	std	Renvoie l'écart type des valeurs des éléments d'une matrice/d'un vecteur ou des éléments le long de l'axe donné
double vector.Var() double matrix.Var() vector matrix.Var(const int axis)	var	Calculer la variance des valeurs des éléments d'une matrice/d'un vecteur
double vector.CorrCoef(const vector& v) matrix matrix.CorrCoef()	corrcoef	Calcule le coefficient de corrélation de Pearson (coefficient de corrélation linéaire). Le coefficient de corrélation est dans la plage [-1, 1]
vector vector.Correlate(const vector& v,enum mode)	correlate	Calcule la corrélation croisée de deux vecteurs. Le paramètre 'mode'

© 2000-2024, MetaQuotes Ltd.

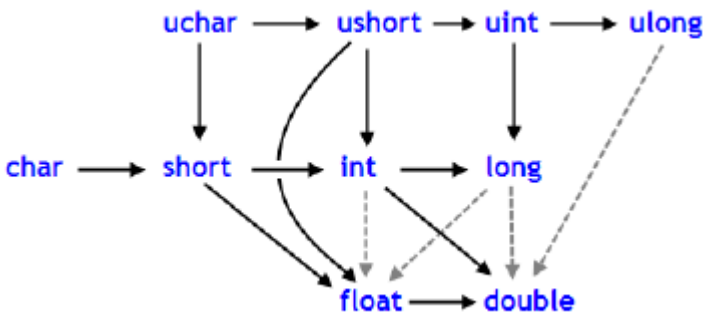
Méthode Matrice/vecteur	Méthode analogue dans NumPy	Description
		détermine le mode de calcul de la convolution linéaire
vector vector.Convolve(const vector& v, enum mode)	convolve	Renvoie la convolution linéaire discrète de deux vecteurs. Le paramètre 'mode' détermine le mode de calcul de la convolution linéaire
matrix matrix.Cov() matrix vector.Cov(const vector& v); (resulting matrix 2 x 2)	cov	Calculer la matrice de covariance. La covariance de deux échantillons (deux variables aléatoires) est une mesure de leur dépendance linéaire

© 2000-2024, MetaQuotes Ltd.

Conversion de Type

Conversion des Types Numériques

Il est souvent nécessaire de convertir un type numérique dans un autre type. Tous les types numériques ne peuvent pas être convertis dans un autre type. Voici le schéma de conversion autorisée :



Les lignes fléchées pleines indiquent les changements qui sont effectués pratiquement sans perte d'information. Au lieu du type char, le type bool peut être utilisé (les deux ne prennent qu'un octet en mémoire), au lieu du type int, le type color peut être utilisé (4 octets), au lieu du type long, datetime peut être utilisé (prend 8 octets). Les 4 lignes grises pointillées, également fléchées, dénotent une conversion lorsque une perte de précision peut se produire. Par exemple, le nombre de chiffres dans l'entier 123456789 (int) est supérieur au nombre de chiffres pouvant être représentés par un float.

```
int n=123456789;
float f=n;    // le contenu de f est égal à 1.234567892E8
Print("n = ",n,"   f = ",f);
// résultat n= 123456789    f= 123456792.00000
```

Un nombre converti en float a le même ordre, mais est moins précis. Les conversions, contrairement aux flèches noires, peuvent être effectuées avec une perte possible d'information. Les conversions entre char et uchar, short et ushort, int et uint, long et ulong (conversions dans les 2 sens) peut amener à une perte de données.

La partie fractionnelle est toujours supprimée lors de la conversion des valeurs à virgule flottante vers un type entier. Si vous souhaitez arrondir un float au nombre entier le plus proche (ce qui est plus pratique dans de nombreux cas), vous devriez utiliser la fonction MathRound().

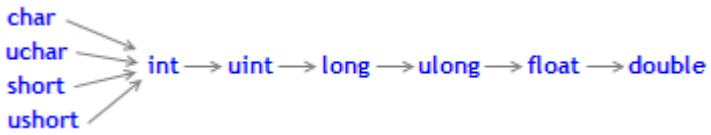
Exemple :

```
//--- Accélération gravitationnelle
   double g=9.8;
   double round_g=(int)g;
   double math_round_g=MathRound(g);
   Print("round_g = ",round_g);
   Print("math_round_g = ",math_round_g);
/*
  Résultat :
   round_g = 9
   math_round_g = 10
```

© 2000-2024, MetaQuotes Ltd.

```
*/
```

Si deux valeurs sont combinées avec un opérateur binaire, l'opérande du type le plus bas est converti dans le type le plus haut avant l'exécution de l'opération selon la priorité donnée dans le schéma ci-desous :



Les types de données char, uchar, short, et ushort inconditionnellement sont convertis vers le type int.

Exemples :

```
   char   c1=3;
//--- Premier exemple
   double d2=c1/2+0.3;
   Print("c1/2 + 0.3 = ",d2);
// Résultat :   c1/2+0.3 = 1.3

//--- Deuxième exemple
   d2=c1/2.0+0.3;
   Print("c1/2.0 + 0.3 = ",d2);
// Résultat :   c1/2.0+0.3 = 1.8
```

L'expression calculée consiste en 2 opérations. Dans le premier exemple, la variable c1 de type char est convertie dans une variable temporaire de type int, car la deuxième opérande dans la division, la constante 2, est du type supérieur int. Le résultat de la division entière 3/2 donne la valeur 1, qui est de type int.

Dans la 2ème opération du premier exemple, la deuxième opérande est la constante 0.3, qui est de type double, le résultat de la première opération est donc converti dans une variable temporaire de type double avec la valeur 1.0.

Dans le deuxième exemple, la variable de type char c1 est convertie dans une variable temporaire de type double, car la deuxième opérande de la division, la constante 2.0, est de type double ; aucun autre conversion n'est faire.

Conversion des Types Numériques

Dans les expressions du langage MQL5, les conversions de type explicites et implicites peuvent être utilisées. La conversion de type explicite est écrit comme suit :

```
var_1 = (type)var_2;
```

Le résultat de l'éxecution d'une expression ou d'une fonction peut être utilisé dans la variable var_2. La notation comme une fonction de la conversion explicite de type est également possible :

```
var_1 = type(var_2);
```

Considérons une conversion explicite de type basée sur le premier exemple.

© 2000-2024, MetaQuotes Ltd.

```
//--- Troisième exemple
   double d2=(double)c1/2+0.3;
   Print("(double)c1/2 + 0.3 = ",d2);
// Résultat :   (double)c1/2+0.3 = 1.80000000
```

Avant que la division soit effectuée, la variable c1 est convertie explicitement vers le type double. La constante entière 2 est maintenant convertie dans la valeur 2.0 de type double, car suite à la conversion, la première opérande a pris le type double. En fait, la conversion explicite de type explicite est une opération unaire.

En outre, lors de la tentative de conversion de type, le résultat peut aller au-delà de l'intervalle autorisé. Dans ce cas, le résultat est tronqué. Par exemple :

```
   char c;
   uchar u;
   c=400;
   u=400;
   Print("c = ",c); // Résultat c=-112
   Print("u = ",u); // Résultat u=144
```

Avant que les opérations (exceptées les affectations) ne soient effectuées, les données sont converties dans le type de priorité maximum. Avant que les affectations ne soient effectuées, les données sont converties dans le type cible.

Exemples :

```
   int    i=1/2;         // aucune conversion de type, le résultat est 0
   Print("i = 1/2  ",i);

   int k=1/2.0;          // l'expression est convertie dans le type double,
   Print("k = 1/2  ",k);  // et est ensuite convertie dans le type int cible, le résul

   double d=1.0/2.0;     // aucune conversion de type, le résultat est 0.5
   Print("d = 1/2.0; ",d);

   double e=1/2.0;       // l'expression est convertie dans le type double,
   Print("e = 1/2.0; ",e);// qui est le même type que le type cible, le résultat est (

   double x=1/2;         // l'expression de type int est convertie vers le type cible c
   Print("x = 1/2; ",x);  // le résultat est 0.0
```

Lors de la conversion du type long/ulong en double, la précision peut être perdue si la valeur entière est supérieure à 9223372036854774784 ou inférieure à -9223372036854774784.

```
void OnStart()
  {
   long l_max=LONG_MAX;
   long l_min=LONG_MIN+1;
//--- définit la valeur entière la plus haute, qui ne perdra pas de précision lors de
   while(l_max!=long((double)l_max))
      l_max--;
//--- définit la plus petite valeur entière, qui ne perdra pas de précision lors de la
```

© 2000-2024, MetaQuotes Ltd.

```
   while(l_min!=long((double)l_min))
      l_min++;
//--- déduit l'intervalle trouvé pour les valeurs entières
   PrintFormat("Lors de la conversion d'une valeur entière en double, elle doit être "
               "dans l'intervalle [%I64d, %I64d]",l_min,l_max);
//--- maintenant, voyons ce qui se passe si la valeur passe en dehors de cet interval
   PrintFormat("l_max+1=%I64d, double(l_max+1)=%.f, ulong(double(l_max+1))=%I64d",
               l_max+1,double(l_max+1),long(double(l_max+1)));
   PrintFormat("l_min-1=%I64d, double(l_min-1)=%.f, ulong(double(l_min-1))=%I64d",
               l_min-1,double(l_min-1),long(double(l_min-1)));
//--- reçoit le résultat suivant
// Lors de la conversion d'une valeur entière en double, elle doit être dans l'interva
// l_max+1=9223372036854774785, double(l_max+1)=9223372036854774800, ulong(double(l_ma
// l_min-1=-9223372036854774785, double(l_min-1)=-9223372036854774800, ulong(double(l_
  }
```

Conversion du Type String

Le type string a la plus haute priorité parmi les types simples. Si l'un des opérandes d'une opération est du type string, la seconde opérande sera donc convertie en string automatiquement. Notez que pour une string, l'addition est possible. La conversion explicit d'une string vers n'importe quel type numérique est permise.

Exemples :

```
   string s1=1.0/8;            // l'expression est convertie du type double,
   Print("s1 = 1.0/8; ",s1);     //  vers le type string,
// le résultat est "0.12500000" (une chaîne de 10 caractères)

   string s2=NULL;             // déinitialisation de la chaîne
   Print("s2 = NULL; ",s2);      // le résultat est une chaîne vide
   string s3="Ticket N"+12345; // l'expression est convertie en string
   Print("s3 = \"Ticket N\"+12345",s3);

   string str1="true";
   string str2="0,255,0";
   string str3="2009.06.01";
   string str4="1.2345e2";
   Print(bool(str1));
   Print(color(str2));
   Print(datetime(str3));
   Print(double(str4));
```

Conversion des Pointeurs de Classes de Base vers des Pointeurs de Classes Dérivées

© 2000-2024, MetaQuotes Ltd.

Les objets d'une classe ouverte générée peuvent également être vues comme des objets de la classe de base correspondante. Ceci conduit à des conséquences intéressantes. Par exemple, en dépit du fait que des objets de classes différentes, générés pour une même classe de base, peut différer significativement les uns des autres, nous pouvons créer une liste chaînée (List) de ces objets, puisqu'ils peuvent être vus comme des objets du type de base. Mais l'inverse n'est pas vrai : des objets de la classe de base ne sont pas automatiquement des objets d'une classe dérivée.

Vous pouvez utiliser une conversion explicite pour convertir des pointeurs de la classe de base en pointeurs d'une classe dérivée. Mais vous devez être totalement sûr de la possibilité d'une telle transformation, car sinon une erreur d'exécution critique se produira et le programme mql5 sera stoppé.

Conversion dynamique avec l'opérateur dynamic_cast operator

La conversion dynamique est effectuée avec l'opérateur dynamic_cast qui ne peut être appliqué que sur des pointeurs de classes. La validation du type est effectué pendant l'exécution. Cela signifie que le compilateur ne vérifie pas le type de donnée appliqué pour la conversion lorsque l'opérateur dynamic_cast est utilisé. Si un pointeur est converti dans un type de données qui n'est pas le type réel d'un objet, le résultat est NULL.

```
dynamic_cast <type-id> ( expression )
```

Le paramètre *type-id* entre crochets doit pointer vers une classe définie auparavant. Contrairement au C++, le type de l'opérande *expression* peut être de n'importe quelle valeur sauf pour void.

Exemple :

```
class CBar { };
class CFoo : public CBar { };

void OnStart()
  {
   CBar bar;
//--- conversion dynamique du type du pointeur *bar vers le pointeur *foo est autorisé
   CFoo *foo = dynamic_cast<CFoo *>(&bar); // pas d'erreur critique
   Print(foo);                             // foo=NULL
//--- la tentative de convertir explicitement une référence d'objet de type Bar vers u
   foo=(CFoo *)&bar;                       // erreur runtime critique
   Print(foo);                             // cette chaîne n'est pas exécutée
  }
```

Voir aussi

Types de Données

© 2000-2024, MetaQuotes Ltd.

Type Void et Constante NULL

Syntaxiquement, le type void est un type fondamental comme les types char, uchar, bool, short, ushort, int, uint, color, long, ulong, datetime, float, double et string. Ce type est utilisé soit pour indiquer qu'une fonction ne retourne aucune valeur, ou, comme paramètre d'une fonction, qu'aucun paramètre n'est attendu.

La variable constante prédéfinie **NULL** est de type *void*. Il peut affecter aux variables de n'importe quel type fondamental sans conversion. La comparaison des variables de type fondamental avec la valeur **NULL** est permise.

Exemple :

```
//--- Si la chaîne n'est pas initialisée, affecte alors notre valeur prédéfinie
if(some_string==NULL) some_string="empty";
```

NULL peut également être comparé aux pointeurs des objets créés avec l'opérateur new.

Voir aussi

Variables, Fonctions

© 2000-2024, MetaQuotes Ltd.

Types définis par l'utilisateur

Le mot-clé typedef en C++ permet de céer des types de données définis par l'utilisateur. Pour cela, spécifiez simplement un nouveau nom du type de données pour un type de données existant. Le nouveau type de données n'est pas créé. Un nouveau pour le type existant est défini à la place. Les types définis par l'utilisateur rendent les applications plus flexibles : il est parfois suffisant de changer les instructions du typedef avec des macros de substitution (#define). Les types définis par l'utilisateur améliorent également la lisibilité du code puisqu'il est possible d'appliquer des noms personnalisés à des types de données standard avec typedef. Le format général de création d'un type défini par l'utilisateur :

```
typedef type nouveau_nom;
```

Ici, *type* signifie tout type de données autorisé, tandis que *nouveau_nom* est un nouveau nom pour le type. Un nouveau nom ne remplace pas le nom d'un type existant, il est seulement ajouté. MQL5 permet de créer des pointeurs de fonctions en utilisant typedef.

Pointeur vers la fonction

Un pointeur de fonction est généralement définit avec le format suivant

```
typedef type_de_resultat_de_la_fonction (*Function_name_type)(list_of_input_paramet
```

où après typedef est écrite la signature de la fonction (nombre et types des paramètres d'entrée, ainsi que le type du résultat retourné par la fonction). Un exemple simple de création et d'utilisation d'un pointeur de fonction se trouve en-dessous :

```
//--- déclare un pointeur vers une fonction acceptant deux paramètres de type int
   typedef int (*TFunc)(int,int);
//--- TFunc est un type, et il est possible de déclarer la variable pointeur de fonct
   TFunc func_ptr; // pointeur vers la fonction
//--- déclare les fonctions correspondant à la description TFunc
   int sub(int x,int y) { return(x-y); }  // soustrait un nombre d'un autre
   int add(int x,int y) { return(x+y); }  // additionne deux nombre
   int neg(int x)       { return(~x);  }  // inverse les bits de la variable
//--- la variable func_ptr peut stocker l'adresse de la fonction pour la déclarer ensu
   func_ptr=sub;
   Print(func_ptr(10,5));
   func_ptr=add;
   Print(func_ptr(10,5));
   func_ptr=neg;           // erreur : neg n'a pas le type int (int,int)
   Print(func_ptr(10));    // erreur : deux paramètres requis
```

Dans cet exemple, la variable *func_ptr* peut recevoir les fonctions *sub* et *add* puisqu'elles ont deux entrées de type int chacune, tel que défini dans le pointeur de fonction *TFunc*. Au contraire, la fonction *neg* ne peut pas être assignée au pointeur func_ptr puisque sa signature est différente.

Arrangement des modèles d'évènements dans l'interface graphique

Les pointeurs de fonctions vous permettent de créer facilement des traitements d'évènements lors de la création d'une interface graphique. Utilisons un exemple de la section CButton pour illustrer la façon

© 2000-2024, MetaQuotes Ltd.

de créer des boutons et ajouter les fonctions de traitement de l'appui sur les boutons. Premièrement, définissons un pointeur vers la fonction *TAction* devant être appelée lors de l'appui sur le bouton et créons trois fonctions selon la *description de TAction*.

```
//--- créons un type de fonction personnalisé
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Ouvre le fichier                                                |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("Fonction %s appelée (nom=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Sauvegarde le fichier                                           |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("Fonction %s appelée (nom=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Ferme le fichier                                                |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("Fonction %s appelée (nom=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
```

Crée ensuite l'instance de classe MyButton de CButton, où nous devons ajouter le pointeur de fonction *TAction*.

```
//+------------------------------------------------------------------+
//| Crée la classe du bouton avec traitement des évènements          |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // handler des évènements du graphic
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- constructeur spécifiant le texte du bouton et le pointeur vers la fonction de
                     MyButton(string text, TAction act)
     {
      Text(text);
      m_action=act;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
   //--- définit la fonction personnalisée appelée depuis le gestionnaire d'évènements
   void              SetAction(TAction act){m_action=act;}
   //--- gestionnaire standard d'évènements du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
      {
       if(m_action!=NULL && lparam==Id())
         {
          //--- appelle le gestionnaire personnalisée m_action()
          m_action(sparam,(int)lparam);a
          return(true);
         }
       else
       //--- retourne le résultat de l'appel au gestionnaire depuis la classe parente (
          return(CButton::OnEvent(id,lparam,dparam,sparam));
      }
  };
```

Créée la classe CControlsDialog dérivée de CAppDialog, ajoute le tableau *m_buttons* pour stocker les boutons de type *MyButton*, ainsi que les méthodes *AddButton(MyButton &button)* et *CreateButtons()*.

```
//+------------------------------------------------------------------+
//| Classe CControlsDialog                                           |
//| Objectif : panneau graphique de gestion de l'application         |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // tableau de boutons
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- ajoute le bouton
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- crée les boutons
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| Crée l'objet CControlsDialog sur le graphique                    |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
   return(CreateButtons());
//---
```

```
  }
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- décalages et espaces
#define INDENT_LEFT                         (11)      // décalage depuis la gauche (en
#define INDENT_TOP                          (11)      // décalage depuis le haut (en p
#define CONTROLS_GAP_X                      (5)       // coordonnée X de l'espace
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y de l'espace
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur du bouton
#define BUTTON_HEIGHT                       (20)      // hauteur du bouton
//--- pour la zone de texte
#define EDIT_HEIGHT                         (20)      // hauteur
//+------------------------------------------------------------------+
//| Crée et ajoute les boutons au panneau CControlsDialog            |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
//--- calcule les coordonnées des boutons
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- ajoute les objets boutons avec les pointeurs de fonctions
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",Save));
   AddButton(new MyButton("Close",Close));
//--- crée le bouton graphiquement
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("Echec de création du bouton %s %d",b.Text(),i);
         return(false);
        }
      //--- ajoute chaque bouton au conteneur CControlsDialog
      if(!Add(b))
         return(false);
     }
//--- succès
   return(true);
  }
```

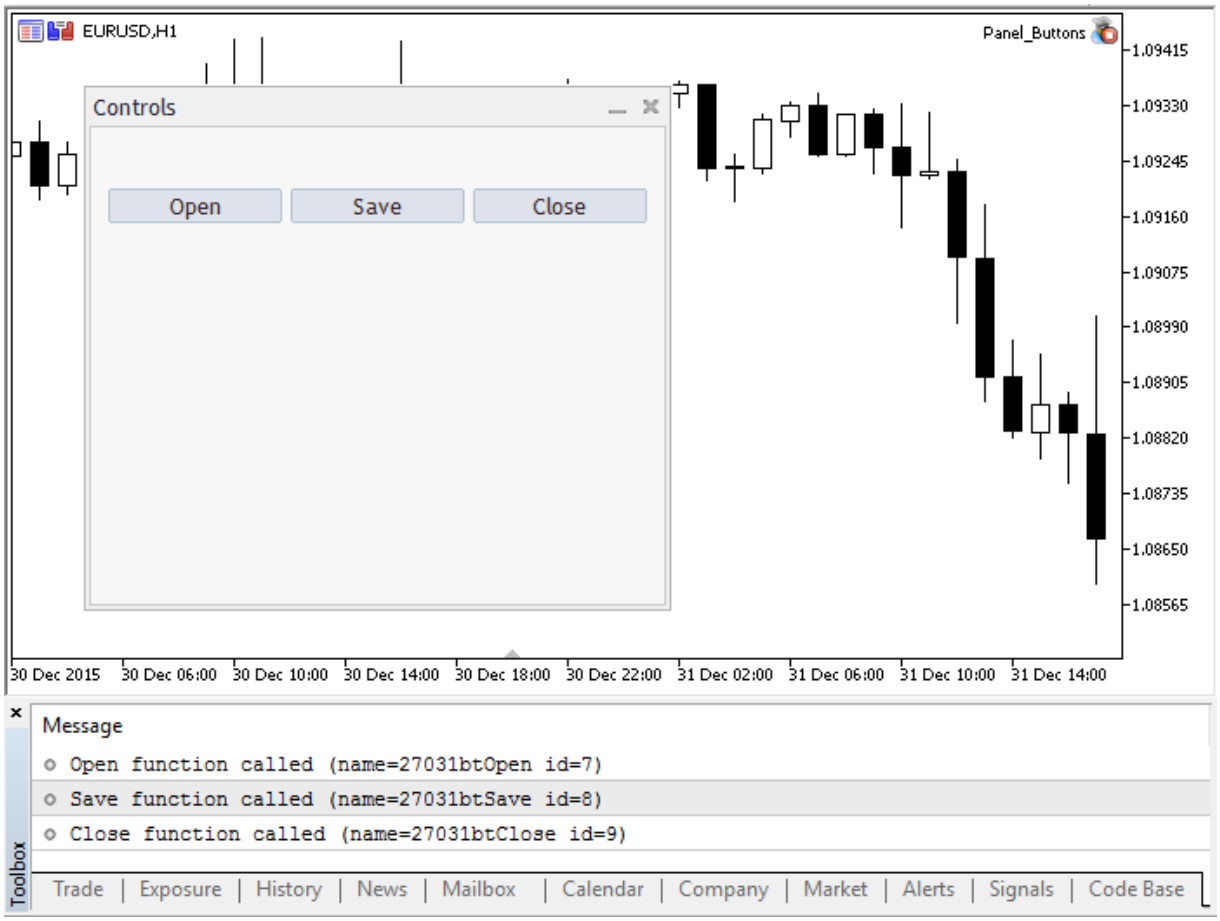
© 2000-2024, MetaQuotes Ltd.

Nous pouvons maintenant développer le programme avec le panneau de contrôle CControlsDialog ayant 3 boutons : Open, Save et Close. En cliquant sur un bouton, la fonction correspondante sous la forme du pointeur *TAction* est appelée.

```
//--- déclare l'objet au niveau global pour le créer automatiquement lors du lancement
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée maintenant l'objet sur le graphique
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   MyDialog.Run();
//--- application initialisée avec succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de dé-initialisation de l'expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- détruit le dialogue
   MyDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique de l'expert                    |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre de l'évènement de type long
                  const double& dparam, // paramètre de l'évènement de type double
                  const string& sparam) // paramètre de l'évènement de type string
  {
//--- appelle le handler depuis la classe parent (ici CAppDialog) pour les évènements
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

L'apparence de l'application lancée et du résultat de l'appui sur le bouton sont fournis dans la copie d'écran.

© 2000-2024, MetaQuotes Ltd.



Le code source complet du programme

```
//+------------------------------------------------------------------+
//|                                                Panel_Buttons.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+

#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneau avec plusieurs boutons CButton"
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- décalages et espaces
#define INDENT_LEFT                         (11)      // décalage depuis la gauche (en
#define INDENT_TOP                          (11)      // décalage depuis le haut (en
#define CONTROLS_GAP_X                      (5)       // coordonnée X de l'espace
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y de l'espace
```

```
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur du bouton
#define BUTTON_HEIGHT                       (20)      // hauteur du bouton
//--- pour la zone de texte
#define EDIT_HEIGHT                         (20)      // hauteur

//--- crée le type personnalisé de fonction
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Ouvre le fichier                                                |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("Fonction %s appelée (nom=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Sauvegarde le fichier                                           |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("Fonction %s appelée (nom=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Ferme le fichier                                                |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("Fonction %s appelée (nom=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
//+------------------------------------------------------------------+
//| Crée la classe du bouton avec traitement des évènements          |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // handler des évènements du graphic
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- constructeur spécifiant le texte du bouton et le pointeur vers la fonction de
                     MyButton(string text,TAction act)
     {
      Text(text);
      m_action=act;
     }
   //--- définit la fonction personnalisée appelée depuis le gestionnaire d'évènements
```

```
   void              SetAction(TAction act){m_action=act;}
   //--- gestionnaire standard d'évènements du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL && lparam==Id())
        {
         //--- appelle le gestionnaire personnalisé
         m_action(sparam,(int)lparam);a
         return(true);
        }
      else
      //--- retourne le résultat de l'appel au gestionnaire depuis la classe parente (
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
//+------------------------------------------------------------------+
//| Classe CControlsDialog                                           |
//| Objectif : panneau graphique de gestion de l'application         |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // tableau de boutons
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- ajoute le bouton
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- crée les boutons
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| Crée l'objet CControlsDialog sur le graphique                    |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
   return(CreateButtons());
//---
  }
//+------------------------------------------------------------------+
//| Crée et ajoute les boutons au panneau CControlsDialog            |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- calcule les coordonnées des boutons
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- ajoute les objets boutons avec les pointeurs de fonctions
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",Save));
   AddButton(new MyButton("Close",Close));
//--- crée le bouton graphiquement
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("Echec de création du bouton %s %d",b.Text(),i);
         return(false);
        }
      //--- ajoute chaque bouton au conteneur CControlsDialog
      if(!Add(b))
         return(false);
     }
//--- succès
   return(true);
  }
//--- déclare l'objet au niveau global pour le créer automatiquement lors du lancement
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée maintenant l'objet sur le graphique
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   MyDialog.Run();
//--- application initialisée avec succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de dé-initialisation de l'expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- détruit le dialogue
   MyDialog.Destroy(reason);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique de l'expert                    |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre de l'évènement de type long
                  const double& dparam, // paramètre de l'évènement de type double
                  const string& sparam) // paramètre de l'évènement de type string
  {
//--- appelle le handler depuis la classe parent (ici CAppDialog) pour les évènements
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Voir aussi

Variables, Fonctions

© 2000-2024, MetaQuotes Ltd.

Pointeurs d'Objets

MQL5 permet la création dynamique d'objets de type complexe. Ceci est fait en utilisant l'opérateur 'new'qui renvoie un descripteur de l'objet créé. La taille du descripteur est de 8 octets. Syntaxiquement, les descripteurs d'objets dans MQL5 sont similaires aux pointeurs C++.

Exemple :

```
MyObject* hobject= new MyObject();
```

Contrairement au C++, la variable "hobject" de l'exemple ci-dessus n'est pas un pointeur vers la mémoire, mais un descripteur d'objet. De plus, en MQL5, tous les objets en paramètres de fonction doivent être passés par référence. Les exemples ci-dessous montrent le passage d'objets en paramètres de fonction :

```
class Foo
  {
public:
   string            m_name;
   int               m_id;
   static int        s_counter;
   //--- constructors and destructors
                     Foo(void){Setup("noname");};
                     Foo(string name){Setup(name);};
                    ~Foo(void){};
   //--- initialise l'objet Foo
   void              Setup(string name)
     {
      m_name=name;
      s_counter++;
      m_id=s_counter;
     }
  };
int Foo::s_counter=0;
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare l'objet comme une variable, avec la création automatique
   Foo foo1;
//--- variante du passage d'un objet par référence
   PrintObject(foo1);

//--- déclare un pointeur sur un objet et le crée avec l'opérateur 'new'
   Foo *foo2=new Foo("foo2");
//--- variante de passage d'un pointeur vers un objet par référence
   PrintObject(foo2); // le pointeur vers l'objet est automatiquement converti par le

//--- déclare un tableau d'objets Foo
```

© 2000-2024, MetaQuotes Ltd.

```
   Foo foo_objects[5];
//--- variante du passage d'un tableau d'objets
   PrintObjectsArray(foo_objects); // une fonction séparée pour passer un tableau d'ob

//--- déclare un tableau de pointeurs vers des objets de type Foo
   Foo *foo_pointers[5];
   for(int i=0;i<5;i++)
      foo_pointers[i]=new Foo("foo_pointer");
//--- variante du passage d'un tableau de pointeurs
   PrintPointersArray(foo_pointers); // une fonction séparée pour passer un tableau de

//--- avant de terminer, assurez-vous de supprimer les objets créés en tant que pointe
   delete(foo2);
//--- supprime le tableau de pointeurs
   int size=ArraySize(foo_pointers);
   for(int i=0;i<5;i++)
      delete(foo_pointers[i]);
//---
  }
//+------------------------------------------------------------------+
//|  Les objets sont toujours passés par référence                   |
//+------------------------------------------------------------------+
void PrintObject(Foo &object)
  {
   Print(__FUNCTION__,": ",object.m_id," Nom de l'objet=",object.m_name);
  }
//+------------------------------------------------------------------+
//| Passe un tableau d'objets                                        |
//+------------------------------------------------------------------+
void PrintObjectsArray(Foo &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
//| Passe un tableau de pointeurs d'objets                           |
//+------------------------------------------------------------------+
void PrintPointersArray(Foo* &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
```

Vérifie le pointeur avant utilisation

© 2000-2024, MetaQuotes Ltd.

Une tentative d'accès à un pointeur invalide provoque l'arrêt critique du programme. La fonction CheckPointer est utilisée pour vérifier un pointeur avant utilisation. Le pointeur peut être invalide dans les cas suivants :

- le pointeur est égal à NULL ;
- l'objet a été détruit à l'aide de l'opérateur delete.

Cette fonction peut être utilisée pour valider un pointeur. Une valeur différente de zéro indique que les données sont accessibles au niveau de ce pointeur.

```
class CMyObject
 {
protected:
  double              m_value;
public:
                      CMyObject(void);
                      CMyObject(double value) {m_value=value;};
                     ~CMyObject(void){};
  //---
  double              Value(void) {return(m_value);}
 };
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- crée un objet non initialisé
  CMyObject *pointer;
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("1. le pointeur est ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

//--- initialise le pointeur
  pointer=new CMyObject(M_PI);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("2. le pointeur est ", EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=", pointer.Value());

//--- supprime l'objet
  delete(pointer);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("3. le pointeur est ", EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=", pointer.Value());
 }
/*
   1. le pointeur  est POINTER_INVALID
   2. pointeur.Value()=3.141592653589793
```

© 2000-2024, MetaQuotes Ltd.

```
   3. le pointeur  est POINTER_INVALID
*/
```

Pour valider rapidement le pointeur, vous pouvez également utiliser "!" (LNOT) qui le vérifie via un appel implicite à la fonction CheckPointer. Cela permet d'écrire du code plus concis et plus clair. Vous trouverez ci-dessous les vérifications de l'exemple précédent :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- crée un objet non initialisé
  CMyObject *pointer;
  if(!pointer)
    Print("1. le pointeur est ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

//--- initialise le pointeur
  pointer=new CMyObject(M_PI);
  if(!pointer)
    Print("2. le pointeur est ", EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=", pointer.Value());

//--- supprime l'objet
  delete(pointer);
  if(!pointer)
    Print("3. le pointeur est ", EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=", pointer.Value());
 }
/*
   1. le pointeur  est POINTER_INVALID
   2. pointeur.Value()=3.141592653589793
   3. le pointeur  est POINTER_INVALID
*/
```

L'opérateur "==" est utilisé pour une vérification rapide de NULL. Par exemple : ptr==NULL ou ptr!=NULL.

Voir aussi

Variables, Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Références : Modificateur & et Mot-Clé this

Passage de Paramètres par Référence

En MQL5, les paramètres des types simples peuvent être passés à la fois par valeur et par référence, tandis que les paramètres des types composés sont toujours passés par référence. Pour informer le compilateur qu'un paramètre doit être passé par référence, le caractère esperluette (ou "et commercial") **&** est ajouté avant le nom du paramètre.

Passer un paramètre par référence signifie passer l'adresse de la variable, c'est pourquoi tous les changements dans un paramètre passé par référence sont immédiatement reflétés dans la variable source. Grâce aux paramètres passés par référence, vous pouvez implémenter le retour de plusieurs résultats d'une fonction en même temps. Afin d'empêcher le changement d'un paramètre passé par référence, utilisez le modificateur const.

Ainsi, si le paramètre d'entrée d'une fonction est un tableau, une structure ou un objet de classe, un symbole '&' est placé dans la signature de la fonction après le type de la variable et avant son nom.

Exemple

```
class CDemoClass
  {
private:
   double            m_array[];

public:
   void              setArray(double &array[]);
  };
//+------------------------------------------------------------------+
//| Remplissage du tableau                                           |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
     ArrayResize(m_array,ArraySize(array));
     ArrayCopy(m_array, array);
     }
  }
```

Dans l'exemple ci-dessus, la classe CDemoClass déclarée contient le membre privé - m_array[] tableau de données de type double. La fonction setArray() est déclarée et prend un tableau array[] passé par référence. Si la signature de la fonction ne contient pas d'indication sur un passage par référence, c'est à dire qu'il ne contient pas le caractère &, un message d'erreur sera généré au moment de la compilation du code.

En dépit du fait que le tableau est passé par référence, il n'est pas possible d'affecter un tableau à un autre. Il faut effectuer la copie du contenu des éléments du tableau source dans le tableau destination un par un. La présence du caractère & dans la signature de la fonction est la condition obligatoire pour les tableaux et les structures passés comme paramètre de fonction.

© 2000-2024, MetaQuotes Ltd.

Mot-Clé this

Une variable de type class (objet) peut être passée à la fois par référence et par pointeur. Comme une référence, le pointeur permet d'avoir accès à un objet. Une fois que le pointeur de l'objet est déclaré, l'opérateur new doit être utilisé pour le créer et l'initialiser.

Le mot-clé réservé **this** permet d'obtenir la référence de l'objet sur lui-même, et est disponible dans les méthodes d'une classe ou d'une structure. **this** référence toujours l'objet dans la méthode dans laquelle il est utilisé et l'expression GetPointer(this) retourne le pointeur de l'objet dans la fonction dans laquelle l'appel est effectué. En MQL5, les fonctions ne peuvent pas retourner d'objet, mais peuvent retourner un pointeur d'objet.

Donc, si nous avons besoin d'une fonction retournant un objet, nous pouvons retourner un pointeur sur cet objet grâce à GetPointer(this). Ajoutons dans la classe CDemoClass une fonction getDemoClass() retournant un pointeur sur l'objet de cette classe.

```
class CDemoClass
  {
private:
   double            m_array[];

public:
   void              setArray(double &array[]);
   CDemoClass       *getDemoClass();
  };
//+------------------------------------------------------------------+
//| Remplissage du tableau                                           |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
      ArrayResize(m_array,ArraySize(array));
      ArrayCopy(m_array,array);
     }
  }
//+------------------------------------------------------------------+
//| retourne son propre pointeur                                     |
//+------------------------------------------------------------------+
CDemoClass *CDemoClass::getDemoClass(void)
  {
   return(GetPointer(this));
  }
```

Les structures n'ont pas de pointeurs, les opérateurs *new* et *delete* ne peuvent pas être appliqués, GetPointer(this) ne peut pas être utilisé.

Voir aussi

Pointeurs d'Objets, Créer et Supprimer des Objets, Portée de Visibilité et Cycle de Vie des Variables

© 2000-2024, MetaQuotes Ltd.

Opérations et Expressions

Certains caractères, ou séquences de caractères, sont importants. Ce sont les symboles des opérations, par exemple :

```
+ - * / %      Symboles des opérations arithmétiques
&& ||          Symboles des opérations logiques
= += *=        Symboles des opérations d'assignement
```

Les symboles d'opérations sont utilisés dans les expressions et ont un sens lorsque les opérandes appropriées sont fournies. Les signes de ponctuation sont également importants. Ce sont les parenthèes, les accolades, les virgules, les deux-points et les points virgules.

Les symboles d'opérations, les marques de ponctuation et les espaces sont utilisés pour séparer les éléments du langages les uns des autres.

Cette section contient la description des éléments suivants :

- Expressions
- Opérations Arithmétiques
- Opérations d'Assignement
- Opérations de Relation
- Opérations Booléennes
- Opérations sur les Bits
- Autres Opérations
- Priorités et Ordre des Opérations

© 2000-2024, MetaQuotes Ltd.

Expressions

Une expression consiste en une ou plusieurs opérandes et symboles d'opérations. Une expression peut être écrite sur plusieurs lignes.

Exemples :

```
a++; b = 10;          // plusieurs expressions sont mises sur une seule ligne
//---  une expression est divisée en plusieurs lignes
x = (y * z) /
    (w + 2) + 127;
```

Une expression qui se termine avec un point-virgule (;) est un opérateur.

Voir aussi

Règles de Précédence

© 2000-2024, MetaQuotes Ltd.

Opérations Arithmétiques

Les opérations arithmétiques incluent les opérations additives et multiplicatives :

```
Somme de variables                          i = j + 2;
Différence des variables                    i = j - 3;
Changement de signe                             x = - x;
Produit des variables                       z = 3 * x;
Division                                    i = j / 5;
Reste de la division                        minutes = heure % 60;
Ajout de 1 à la valeur de la variable           i++;
Ajout de 1 à la valeur de la variable            ++i;
Soustraction de 1 à la valeur de la variable     k--;
Soustraction de 1 à la valeur de la variable     --k;
```

Les opérations d'incrémentation et de décrémentation ne sont appliquées qu'aux variables, elles ne peuvent pas être appliquées aux constantes. Les préfixes d'incrémentation (++i) et de décrémentation (--k) sont appliqués aux variables juste avant l'utilisation de cette variable dans une expression.

La post-incrémentation (i++) et la post-décrémentation (k--) sont appliqués à la variable juste après l'utilisation de cette variable dans une expression.

Note Importante

```
int i=5;
int k = i++ + ++i;
```

Des problèmes de calcul peuvent se produire lors du déplacement de l'expression ci-dessus d'un environnement de programmation vers un autre (par exemple, de Borland C++ à MQL5). En général, l'ordre des calculs dépend de l'implémentation du compilateur. En pratique, il y a deux façons d'implémenter la post-décrémentation (post-incrémentation) :

1. La post-décrémentation (post-incrémentation) est appliquée à la variable après avoir calculé l'expression entière.
2. La post-décrémentation (post-incrémentation) est appliquée à la variable immédiatement au moment de l'opération.

La première méthode de calcul de la post-décrémentation (post-incrémentation) est actuellement implémentée en MQL5. Mais même en sachant cette particularité, il n'est pas recommandé d'expérimenter son usage.

Exemples :

```
int a=3;
a++;            // expression valide
int b=(a++)*3;  // expression invalide
```

Voir aussi

Règles de Précédence

© 2000-2024, MetaQuotes Ltd.

Opérations d'Assignement

La valeur de l'expression qui inclut l'opération donnée est la valeur de l'opérande de gauche après l'assignement :

```
Assigner la valeur de x à la variable y                                  y = x;
```

Les opérations suivantes unissent les opérations arithmétiques ou sur les bits avec l'opération d'affectation :

```
Ajout de x à la variable y                                               y += x;
Soustraction de x de la variable y                                       y -= x;
Multiplication de la variable y par x                                    y *= x;
Division de la variable y par x                                          y /= x;
Reste de la division de la variable y par x                              y %= x;
Décalage de la représentation binaire de y vers la droite de x bits      y >>= x;
Décalage de la représentation binaire de y vers la gauche de x bits      y <<= x;
Opération binaire AND des représentations binaires de y et de x          y &= x;
Opération binaire OU des représentations binaires de y et de x           y |= x;
Opération binaire OU Exclusif des représentations binaires de y et de x  y ^= x;
```

Les opérations sur les bits ne peuvent être effectuées sur les entiers. Lorsque des opérations de décalage logique de la représentation de y vers la droite/gauche de x bits sont effectuées, les 5 plus petits chiffres binaires de la valeur de x sont utilisés, les plus grands sont ignorés, c'est à dire que le décalage est effectué sur les bits 0-31.

Pour l'opération %= (y modulo x), le signe du résultat est égal au signe du nombre divisé.

L'opérateur d'assignement peut être utilisé plusieurs fois dans une expression. Dans ce cas, le traitement de l'expression est effectué de gauche à droite :

```
y=x=3;
```

Premièrement, la valeur 3 sera affectée à la variable x, la valeur de x, c'est à dire 3, sera ensuite affectée à la variable y.

Voir aussi

Règles de Précédence

© 2000-2024, MetaQuotes Ltd.

Opérations de Relation

Le booléen FAUX (FALSE) est représenté avec la valeur entière zéro, tandis que le booléen VRAI (TRUE) est représenté par n'importe quelle valeur différente de zéro.

La valeur des expressions contenant des opérations de relation ou des opérations logiques est FALSE (0) ou TRUE (1).

```
True si a est égal à b                    a == b;
True si a n'est pas égal à b              a != b;
True si a est inférieur à b               a < b;
True si a est supérieur à b               a > b;
True si a est inférieur ou égal à b       a <= b;
True si a est supérieur ou égal à b       a >= b;
```

Deux nombres réels ne peuvent pas être comparés. Dans la plupart des cas, deux chiffres semblant identiques peuvent être inégaux en raison de valeurs différentes à la 15ème décimale. Pour pouvoir comparer correctement deux nombres réels, comparez la différence normalisée de ces nombres avec zéro.

Exemple :

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0) return(true);
   else return(false);
  }
void OnStart()
  {
   double first=0.3;
   double second=3.0;
   double third=second-2.7;
   if(first!=third)
     {
      if(CompareDoubles(first,third))
         printf("%.16f et %.16f sont égaux",first,third);
     }
  }
// Résultat : 0.3000000000000000  0.2999999999999998   sont égaux
```

Voir aussi

Règles de Précédence

© 2000-2024, MetaQuotes Ltd.

Opérations Booléennes

Négation Logique NON (!)

L'opérande de la négation logique (!) doit être de type arithmétique. Le résultat est TRUE (1) si la valeur de l'opérande est FALSE (0) ; et il est égal à FALSE (0) si l'opérande diffère de FALSE (0).

```
if(!a) Print("not 'a'");
```

Opération Logique OU (||)

Opération logique OU (||) des valeurs x et y. La valeur de l'expression est TRUE (1) si la valeur de x ou y est vraie (non null). Sinon - FALSE (0).

```
if(x<0 || x>=max_bars) Print("en dehors du range");
```

Opération Logique AND (&&)

Opération logique ET (&&) des valeurs x et y. La valeur de l'expression est TRUE (1) si les valeurs de x et y sont vraies (non null). Sinon - FALSE (0).

Brève Estimation des Opérations Booléennes

Le schéma de la fameuse "brève estimation" est appliqué aux opérations booléennes, c'est à dire que le calcul de l'expression est terminé quand le résultat de l'expression peut être estimé avec précision.

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- premier exemple d'estimation brève
   if(func_false() && func_true())
     {
      Print("Opération && : Vous ne verrez jamais cette expression");
     }
   else
     {
      Print("Opération && : le résultat de la première expression est faux, la deuxièr
     }
//--- deuxième exemple d'estimation brève
   if(!func_false() || !func_true())
     {
      Print("Opération || : le résultat de la première expression est vrai, la deuxièr
     }
   else
     {
      Print("Opération || : Vous ne verrez jamais cette expression");
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| la fonction retourne toujours false                              |
//+------------------------------------------------------------------+
bool func_false()
  {
   Print("Fonction func_false()");
   return(false);
  }
//+------------------------------------------------------------------+
//| la fonction retourne toujours true                               |
//+------------------------------------------------------------------+
bool func_true()
  {
   Print("Fonction func_true()");
   return(true);
  }
```

Voir aussi

Règles de Précédence

© 2000-2024, MetaQuotes Ltd.

Opérations sur les Bits

Complément à Un

Complément de la variable jusqu'à un. La valeur de l'expression contient 1 pour tous les chiffres où la valeur de la variable contient 0, et 0 pour tous les chiffres où la variable contient 1.

```
b = ~n;
```

Exemple :

```
   char a='a',b;
   b=~a;
   Print("a = ",a, "  b = ",b);
// Le résultat sera :
// a = 97   b = -98
```

Décalage vers la Droite

La représentation binaire de x est décalée vers la droite de y chiffres. Si la valeur à décaler est d'un type non signé, le décalage logique vers la droite est effectué, c'est à dire que les bits libres du côté gauche seront remplis avec des zéros.

Si la valeur à décaler est d'un type signé, le décalage arithmétique vers la droite est effectué, c'est à dire que les bits libres du côté gauche seront remplis avec la valeur du bit signé (si le nombre est positif, la valeur du bit signé est 0 ; si le nombre est négatif, la valeur du signe du bit est 1).

```
x = x >> y;
```

Exemple :

```
   char a='a',b='b';
   Print("Avant :  a = ",a, "  b = ",b);
//--- décalage vers la droite
   b=a>>1;
   Print("Après :   a = ",a, "  b = ",b);
// Le résultat sera :
// Avant :  a = 97   b = 98
// Après :   a = 97   b = 48
```

Décalage vers la Gauche

La représentation binaire de x est décalée vers la gauche de y chiffres, les chiffres libres du côté droit sont remplis avec des zéros.

```
x = x << y;
```

Exemple :

```
   char a='a',b='b';
   Print("Avant :  a = ",a, "  b = ",b);
//--- décalage vers la gauche
   b=a<<1;
```

© 2000-2024, MetaQuotes Ltd.

```
   Print("Après :   a = ",a, "  b = ",b);
// Le résultat sera :
// Avant :  a = 97   b = 98
// Après :   a = 97   b = -62
```

Il n'est pas recommandé de décalé d'un nombre de bits plus grand ou égal à la longueur de la variable décalée, car le résultat de ce type d'opération n'est pas défini.

Opération AND sur les Bits

L'opération AND sur les bits des représentations binaires x et y. La valeur de l'expression contient 1 (TRUE) dans tous les chiffres où à la fois x et y ne contiennent pas 0, et il contient 0 (FALSE) dans tous les autres chiffres.

```
b = ((x & y) != 0);
```

Exemple :

```
   char a='a',b='b';
//--- Opération AND
   char c=a&b;
   Print("a = ",a,"  b = ",b);
   Print("a & b = ",c);
// Le résultat sera :
// a = 97   b = 98
// a & b = 96
```

Opération OU sur les Bits

L'opération OU sur les bits des représentations binaires x et y. La valeur de l'expression contient 1 dans tous les chiffres où x ou y ne contiennent pas 0, et il contient 0 dans tous les autres chiffres.

```
b = x | y;
```

Exemple :

```
   char a='a',b='b';
//--- Opération OR
   char c=a|b;
   Print("a = ",a,"  b = ",b);
   Print("a | b = ",c);
// Le résultat sera :
// a = 97   b = 98
// a | b = 99
```

Opération OU Exclusif de bits

L'opération OU exclusif (eXclusive OR) sur les bits des représentations binaires de x et y. La valeur de l'expression contient 1 dans tous les chiffres où x et y ont des valeurs binaires différentes, et il contient 0 dans tous les autres chiffres.

```
b = x ^ y;
```

© 2000-2024, MetaQuotes Ltd.

Exemple :

```
   char a='a', b='b';
//--- Opération OU Exclusif
   char c=a^b;
   Print("a = ",a,"  b = ",b);
   Print("a ^ b = ",c);
// Le résultat sera :
// a = 97   b = 98
// a ^ b = 3
```

Les opérations sur les bits ne sont effectués qu'avec des entiers.

Voir aussi

Règles de Précédence

© 2000-2024, MetaQuotes Ltd.

Autres opérations

Indexation ([])

Lors de l'adressage du i-ème élément du tableau, la valeur de l'expression est la valeur d'une variable avec le numéro de série i.

Exemple :

```
array[i] = 3; // Assigne la valeur 3 au i-ème élément du tableau.
```

Seul un entier peut servir d'indice dans un tableau. Les tableaux jusqu'à 4 dimensions sont autorisés. Chaque dimension est indexée de 0 à **taille de la dimension**-1. En particulier, pour un tableau à 1 dimension constitué de 50 éléments, la référence au premier élément sera array[0], celui du dernier élément sera array[49].

Lors d'un adressage au délà du tableau, le sous-système d'exécution générera une erreur critique et le programme sera stoppé.

Appel d'une Fonction avec les Arguments x1, x2 ,..., xn

Chaque argument peut représenter une constante, une variable ou une expression du type correspondant. Les arguments passés sont séparés par des virgules et doivent être entre parenthèses, la parenthèse ouvrante doit suivre le nom de la fonction appelée.

La valeur de l'expression est la valeur retournée par l'expression. Si la valeur de retour est de type void, ce type d'appel de fonction ne peut pas être placé à la droite de l'opération d'affectation. Notez que les expressions x1,..., xn sont exécutées exactement dans cet ordre.

Exemple :

```
   int length=1000000;
   string a="a",b="b",c;
//--- Autres Opérations
   int start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("heure pour 'c = a + b' = ",(stop-start)," millisecondes, i = ",i);
```

Opérateur Virgule (,)

Les expressions séparées par des virgules sont exécutéers de gauche à droite. Tous les effets de bord du calcul de l'expression de gauche peuvent apparaître avant que l'expression de droite ne soit calculée. Le type du résultat et sa valeur coïncide avec ceux de l'expression de droite. La liste des paramètres à passer (voir ci-dessus) peut être considérée comme un exemple.

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
for(i=0,j=99; i<100; i++,j--) Print(array[i][j]);
```

Operateur (.)

Pour accéder directement aux membres publics des structures et des classes, l'opération point (.) est utilisée. Syntaxe :

```
Nom_variable_de_type_structure.Nom_membre
```

Exemple :

```
struct SessionTime
  {
   string sessionName;
   int    startHour;
   int    startMinutes;
   int    endHour;
   int    endMinutes;
  } st;
st.sessionName="Asian";
st.startHour=0;
st.startMinutes=0;
st.endHour=9;
st.endMinutes=0;
```

Opérateur de Résolution de Portée (::)

Chaque fonction dans un programme mql5 a sa propre portée d'exécution. Par exemple, la fonction système Print() est effectuée dans une portée globale. Les fonctions importées sont appelées dans la portée de l'import correspondant. Les fonctions des classes ont la portée de la classe correspondante. La syntaxe de l'opération de résolution de la portée est comme suit :

```
[Nom_portée]::Nom_fonction(paramètres)
```

S'il n'y a pas de nom de portée, c'est une indication explicite d'utiliser la portée globale. S'il n'y a pas d'opération de résolution de portée, la fonction est associée à la portée la plus proche. S'il n'y a pas de fonction dans la portée locale, la recherche est conduite dans la portée globale.

L'opération de résolution de portée est également utilisée pour définir les fonctions membres de classe.

```
type Nom_classe::Nom_fonction(description_parametres)
   {
// corps de la fonction
   }
```

L'utilisation de plusieurs fonctions du même nom dans différents contextes d'exécution dans un programme peuvent causer des ambiguïtés. L'ordre de priorité des appels de fonctions sans spécification de portée explicite est le suivant :

1. Méthodes de classe. Si aucune fonction avec le nom spécifié n'existe dans la classe, va au niveau suivant.

© 2000-2024, MetaQuotes Ltd.

2. Fonctions MQL5. Si le langage n'a pas la fonction demandée, va au niveau suivant.
3. Fonctions globales utilisateur Si aucune fonction avec le nom spécifié n'est trouvée, va au niveau suivant.
4. Fonctions importées. Si aucune fonction avec le nom spécifié n'est trouvée, le compilateur retourne une erreur.

Pour éviter l'ambiguïté des appels de fonction, spécifiez toujours explicitement la portée de la fonction en utilisant l'opération de résolution de portée.

Exemple :

```
#property script_show_inputs
#import "kernel32.dll"
   int GetLastError(void);
#import

class CCheckContext
  {
   int         m_id;
public:
               CCheckContext() { m_id=1234; }
protected:
   int         GetLastError() { return(m_id); }
  };
class CCheckContext2 : public CCheckContext
  {
   int         m_id2;
public:
               CCheckContext2() { m_id2=5678; }
   void        Print();
protected:
   int         GetLastError() { return(m_id2); }
  };
void CCheckContext2::Print()
  {
   ::Print("Terminal GetLastError",::GetLastError());
   ::Print("kernel32 GetLastError",kernel32::GetLastError());
   ::Print("parent GetLastError",CCheckContext::GetLastError());
   ::Print("notre GetLastError",GetLastError());
  }
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CCheckContext2 test;
   test.Print();
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
```

Opération d'Obtgention de la Taille du Type de Données ou la Taille d'un Objet de n'Importe Quel Type de Données (sizeof)

En utilisant l'opération sizeof, la taille de la mémoire correspondant à un identifiant ou à un type peut être défini. L'opération sizeof est au format suivant :

Exemple :

```
sizeof(expression)
```

N'importe quel identifiant, ou nom de type entre parenthèses, peut être utilisé comme une expression. Notez que le nom du type void ne peut pas être utilisé, et l'identifiant ne peut pas appartenir au champ de bits ou être un nom de fonction.

Si l'expression est le nom d'un tableau statique (c'est à dire que la première dimension est donnée), alors le résultat est la taille du tableau entier (c'est à dire le prodit du nombre d'éléments et de la longueur du type). Si l'expression est le nom d'un tableau dynamique (la première dimension n'est pas spécifiée), la résultat sera la taille de l'objet du tableau dynamique.

Lorsque sizeof est appliqué au nom d'une structure ou d'une classe, ou à l'identifiant de la structure ou de la classe, le résultat est la taille réelle de la structure ou de la classe.

Exemple :

```
struct myStruct
  {
   char   h;
   int    b;
   double f;
  } str;
Print("sizeof(str) = ",sizeof(str));
Print("sizeof(myStruct) = ",sizeof(myStruct));
```

La taille est calculée au moment de la compilation.

Voir aussi

Règles de Précédence

© 2000-2024, MetaQuotes Ltd.

Règles de Précédence

Chaque groupe d'opérations du tableau a la même priorité. Plus les opérations sont prioritaires, plus son groupe est placé en haut du tableau. Les règles de précédence déterminent le regroupement des opérations et des opérandes.

Attention : La précédence des opérations dans le langage MQL5 correspond à la priorité adoptée en C++, et est différente de la priorité donnée dans le langage MQL4.

Opération	Description	Ordre d'Exécution
() [] .	Appel de Fonction Référencement d'un élément d'un tableau Référencement d'un élément d'une structure	De gauche à droite
! ~ - ++ -- (type) sizeof	Négation logique Négation de bit (complément) Changement de signe Incrément de 1 Décrement de 1 Conversion de Type Détermination de la taille en octets	De droite à gauche
* / %	Multiplication Division Modulo	De gauche à droite
+ -	Addition Soustraction	De gauche à droite
<< >>	Décalage vers la gauche Décalage vers la droite	De gauche à droite
< <= > >=	Inférieur à Inférieur ou égal Supérieur à Supérieur ou égal	De gauche à droite
== !=	Egal à Différent de	De gauche à droite
&	Opération ET de bits	De gauche à droite
^	OU exclusif de bits	De gauche à droite
|	Opération OU de bits	De gauche à droite
&&	Opération ET logique	De gauche à droite
||	Opération OU logique	De gauche à droite
?:	Opérateur Conditionnel	De droite à gauche
=	Assignement	De droite à gauche

© 2000-2024, MetaQuotes Ltd.

Opération	Description	Ordre d'Exécution
*= /= %= += -= <<= >>= &= ^= |=	Multiplication avec affectation Division avec affectation Modulo avec affectation Addition avec affectation Soustraction avec affectation Décalage à gauche avec affectation Décalage à droite avec affectation ET de bits avec affectation OU exclusif de bits avec affectation OU de bits avec affectation	
,	Virgule	De gauche à droite

Pour changer l'ordre d'exécution des opérations, les parenthèses de plus haute priorité sont utilisées.

© 2000-2024, MetaQuotes Ltd.

Opérateurs

Les opérateurs du langage décrivent les opérations algorithmiques qui doivent être exécutées pour accomplir une tâche. Le corps du programme est une suite de tels opérateurs. Les opérateurs qui se suivent sont séparés par des points-virgules.

Opérateur	Description
Opérateur composé {}	Un ou plusieurs opérateurs de tout type, entourés d'accolades {}
Opérateur d'expression (;)	Toute expression qui se termine par un point-virgule (;)
Opérateur return	Termine la fonction courante et rend le contrôle au programme appelant
Opérateur conditionnel if-else	Est utilisé lorsqu'il est nécessaire d'effectuer un choix
Opérateur conditionnel ?:	Une simple analogie de l'opérateur conditionnel if-else
Opérateur de sélection switch	Passe le contrôle à l'opérateur correspondant à la valeur de l'expression
Opérateur de boucle while	Effectue une opération jusqu'à ce que l'expression vérifiée soit fausse. L'expression est vérifiée avant chaque itération
Opérateur de boucle for	Effectue une opération jusqu'à ce que l'expression vérifiée soit fausse. L'expression est vérifiée avant chaque itération
Opérateur de boucle do-while	Effectue une opération jusqu'à ce que l'expression vérifiée soit fausse. La condition de fin est vérifiée après chaque boucle. Le corps de la boucle est toujours effectué au moins une fois.
Opérateur break	Termine l'exécution de l'opérateur switch, while, do-while ou for
Opérateur continue	Passe le contrôle au début de l'itération suivante de l'opérateur while, do-while ou for
Opérateur new	Crée un objet de la taille désirée et retourne un descripteur de l'objet créé.
Opérateur delete	Supprime un objet créé avec l'opérateur new

Un opérateur peut occuper une ou plusieurs lignes. Deux opérateurs ou plus peuvent être situés sur la même ligne. Les opérateurs ayant un contrôle sur l'ordre d'exécution (if, if-else, switch, while et for) peuvent être imbriqués les uns dans les autres.

Exemple :

```
if(Month() == 12)
  if(Day() == 31) Print("Happy New Year!");
```

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur Composé

Un opérateur composé (un bloc) consisten en un ou plusieurs opérateur de n'importe quel type, entouré d'accolades {}. L'accolade fermante ne doit pas être suivie d'un point-virgule (;).

Exemple :

```
if(x==0)
  {
   Print("position invalide x = ",x);
   return;
  }
```

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur d'expression

Toute expression qui se termine par un point-virgule (;), est un opérateur. Voici quelques exemples d'opérateurs d'expression.

Opérateur d'attribution :

L'identificateur est = une expression ;

```
x=3;
y=x=3;
bool equal=(x==y);
```

L'opérateur d'affectation peut être utilisé plusieurs fois dans une expression. Dans ce cas, l'expression est traitée de droite à gauche.

Opérateur d'appel de fonction :

Nom_de fonction (argument1,..., argumentN);

```
FileClose(file);
```

Opérateur Vide

Il se compose seulement d'un point-virgule (;) et est utilisé pour dénoter le corps vide d'un opérateur de contrôle.

Voir aussi

Initialisation des variables, Portée de visibilité et durée de vie des variables, Création et destruction des objets

© 2000-2024, MetaQuotes Ltd.

Opérateur Return

L'opérateur return termine l'exécution de la fonction courante et retourne le contrôle au programme appelant. Le résultat du calcul de l'expression est retourné à la fonction appelante. L'expression peut contenir un opérateur d'affectation.

Exemple :

```
int CalcSum(int x, int y)
  {
   return(x+y);
  }
```

Dans les fonctions ayant un type de retour de type void, l'opérateur return sans expression doit être utilisé :

```
void SomeFunction()
  {
   Print("Hello!");
   return;    // cet opérateur peut être supprimé
  }
```

L'accolade fermante de la fonction signifie l'exécution implicite de l'opérateur return sans expression.

Ce qui peut être retourné : des types simples, des structures simples, des pointeurs d'objet. Vous ne pouvez pas retourner de tableaux, d'objets de type classe et de variables de type structure composée avec l'opérateur *return*.

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur Conditionnel If-Else

L'opérateur IF - ELSE est utilisé lorsqu'un choix doit être fait. Formellement, la syntaxe est la suivante :

```
if (expression)
    opérateur1
else
    opérateur2
```

Si l'expression est vraie, operator1 est exécuté et le contrôle est donné à l'opérateur qui suit operator2 (operator2 n'est pas exécuté). Si l'expression est fausse, operator2 est exécuté.

La partie else de l'opérateur if peut être omise. Une divergence peut donc apparaître dans des opérateurs if imbriqués sans la partie else. Dans ce cas, else s'adresse à l'opérateur if précédent le plus proche dans le même bloc qui n'a pas de partie else.

Exemples :

```
//--- La partie else se réfère au deuxième opérateur if :
if(x>1)
   if(y==2) z=5;
else     z=6;
//--- La partie else se réfère au premier opérateur if :
if(x>1)
  {
   if(y==2) z=5;
  }
else        z=6;
//--- Opérateurs imbriqués
if(x=='a')
  {
   y=1;
  }
else if(x=='b')
  {
   y=2;
   z=3;
  }
else if(x=='c')
  {
   y=4;
  }
else Print("ERREUR");
```

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur Ternaire ?:

La forme générale de l'opérateur ternaire est la suivante :

```
expression1 ? expression2 : expression3
```

Pour la première opérande - "expression1" - toute expression qui résulte en une valeur de type bool peut être utilisée. Si le résultat est vrai (true), alors l'opérateur définit par la seconde opérande, c'est à dire "expression2" est exécutée.

Si la première opérande est fausse (false), la troisième opérande - "expression3" est effectuée. Les seconde et troisième opérande, c'est à dire "expression2" et "expression3" doivent retourner des valeurs du même type et ne doivent pas retourner void type. Le résultat de l'exécution de l'opérateur conditionnel est le résultat de expression2 ou le résultat de expression3, suivant le résultat de expression1.

```
//--- différence normalisée entre les prix open et close pour le range d'une journée
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

Cette entrée est équivalente à :

```
   double true_range;
   if(High==Low)true_range=0;                // si High et Low sont égaux
   else true_range=(Close-Open)/(High-Low); // si le range n'est pas nul
```

Utilisation de l'Opérateur et Restrictions

Suivant la valeur de "expression1", l'opérateur doit retourner l'une des deux valeurs - soit "expression2" ou "expression3". Il y a plusieurs limitations à ces expressions :

1. Ne mélangez pas un type utilisateur avec un type simple ou avec une énumération. NULL peut être utilisé comme pointeur.
2. Si les types des valeurs sont simples, l'opérateur sera du type maximum (voir Conversion de Type).
3. Si l'une des valeurs est un énumération et que la seconde est d'un type numérique, l'énumération est remplacé par int et la deuxième règle est appliquée.
4. Si les deux valeurs sont des énumérations, leurs types doivent être identiques et l'opérateur sera de type énumération.

Restrictions pour les types utilisateurs (classes ou structures) :

a) Les types doivent être identiques ou l'un doit dériver de l'autre.

b) Si les types ne sont pas identiques (héritage), alors l'enfant est implicitement converti dans le type du parent, c'est à dire que l'opérateur sera du type du parent.

c) Ne mélangez pas un objet et un pointeur - les deux expressions doivent être soit des objets, soit des pointeurs. NULL peut être utilisé pour un pointeur.

Note

Faites attention lors de l'utilisation de l'opérateur conditionnel comme argument d'une fonction surchargée, car le type du résultat d'un opérateur conditionnel est défini au moment de la compilation

© 2000-2024, MetaQuotes Ltd.

du programme. Et ce type est déterminé comme étant le plus grand des types de "expression2" et "expression3".

Exemple :

```
void func(double d) { Print("argument double : ",d); }
void func(string s) { Print("argument string : ",s); }

bool   Expression1=true;
double Expression2=M_PI;
string Expression3="3.1415926";

void OnStart()
  {
   func(Expression2);
   func(Expression3);

   func(Expression1?Expression2:Expression3);   // avertissement sur le cast implicite
   func(!Expression1?Expression2:Expression3);  // avertissement sur le cast implicite
  }

//   Résultat :
//   double argument: 3.141592653589793
//   string argument: 3.1415926
//   string argument: 3.141592653589793
//   string argument: 3.1415926
```

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur Switch

Compare la valeur de l'expression avec les constantes de toutes les variantes de type *case* et passe le contrôle à l'opérateur correspondant à la valeur de l'expression. Chaque variante de *case* peut être marquée avec une constante entière, une constante littérale ou une expression constante. L'expression constante ne peut pas contenir de variables ou d'appels de fonctions. L'expression de l'opérateur *switch* doit être de type entier - int or uint.

```
switch(expression)
  {
   case constante: opérateurs
   case constante: opérateurs
      ...
   default: opérateurs
  }
```

Les opérateurs marqués avec l'étiquette *default* sont exécutés si aucune des constantes des opérateurs *case* n'est égale à la valeur de l'expression. La variante *default* ne doit pas nécessairement être déclarée, ni être la dernière. Si aucune des constantes ne correspond à la valeur de l'expression et que la variante *default* n'est pas disponible, aucune action n'est exécutée.

Les mots-clés *case* avec une constante ne sont que des constantes et si les opérateurs sont exécutés pour certaines variantes *case*, le programme exécutera ensuite les opérateurs de toutes les variantes suivantes jusqu'à ce que l'opérateur *break* soit rencontré. Cela permet de lier une séquence d'opérateurs avec plusieurs variantes.

Une expression constante est calculée pendant la compilation. Il n'est pas possible d'avoir deux constantes identiques dans un opérateur *switch*.

Exemples :

```
//--- Premier exemple
switch(x)
  {
   case 'A':
      Print("CASE A");
      break;
   case 'B':
   case 'C':
      Print("CASE B ou C");
      break;
   default:
      Print("NI A, B ou C");
      break;
  }

//---  Deuxième exemple
   string res="";
   int i=0;
   switch(i)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      case 1:
         res=i;break;
      default:
         res="default";break;
      case 2:
         res=i;break;
      case 3:
         res=i;break;
     }
   Print(res);
/*
  Résultat
   default
*/
```

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur de Boucle While

L'opérateur while consiste en l'opérateur et une expression vérifiée qui doit être vérifiée :

```
while(expression)
 operateur;
```

Si l'expression est vraie, l'opérateur est exécuté jusqu'à ce que l'expression devienne fausse. Si l'expression est fausse, le contrôle est passé à l'opérateur suivant. La valeur de l'expression est définie avant que l'opérateur ne soit exécuté. Donc si l'expression est fausse depuis le début, l'opérateur ne sera pas exécuté du tout.

Note

S'il est attendu qu'un grand nombre d'itérations seront gérées dans une boucle, il est conseillé de vérifier que la fin du programme n'a pas été demandée avec la fonction IsStopped().

Exemple :

```
while(k<n && !IsStopped())
  {
   y=y*x;
   k++;
  }
```

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur de Boucle For

L'opérateur for consiste en trois expressions et un opérateur exécutable :

```
for(expression1; expression2; expression3)
  operateur;
```

Expression1 décrit l'initialisation de la boucle. Expression2 vérifie les conditions de fin de la boucle. Si elle est vraie, le corps de la boucle for est exécuté. La boucle répète expression2 jusqu'à ce qu'elle deviennent fausse. Si elle est fausse, la boucle est terminée et le contrôle est donné à l'opérateur suivant. Expression3 est calculée après chaque itération.

L'opérateur for est équivalent à la succession d'opérateurs suivante :

```
expression1;
while(expression2)
  {
  operateur;
   expression3;
  };
```

N'importe quelle des trois ou les trois expressions peuvent être absentes dans l'opérateur for operator, mais les points-virgules (;) les séparant doivent être présents. Si expression2 est omise, elle est constamment considérée comme vraie. L'opérateur for(;;) est une boucle continue, équivalent à l'opérateur while(1). Chaque expression 1 ou 3 peut consister en plusieurs expressions combinées avec un opérateur virgule ','.

Note

S'il est attendu qu'un grand nombre d'itérations seront gérées dans une boucle, il est conseillé de vérifier que la fin du programme n'a pas été demandée avec la fonction IsStopped().

Exemples :

```
for(x=1;x<=7000; x++)
  {
   if(IsStopped())
      break;
   Print(MathPower(x,2));
  }
//--- Autre exemple
for(;!IsStopped();)
  {
   Print(MathPower(x,2));
   x++;
   if(x>10) break;
  }
//--- Troisième exemple
for(i=0,j=n-1;i<n && !IsStopped();i++,j--) a[i]=a[j];
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur de Boucle do while

Les boucles for et while vérifient la condition au début, et non à la fin d'une boucle. Le troisième opérateur de boucle do - while vérifie la condition à la fin, après chaque itération de boucle. Le corps de la boucle est toujours exécuté au moins une fois.

```
do
  operateur;
while(expression);
```

L'opérateur est d'abord exécuté, et l'expression est ensuite calculée. Si elle est vraie, l'opérateur est encore exécutée, et ainsi de suite. Si l'expression devient fausse, la boucle se termine.

Note

S'il est attendu qu'un grand nombre d'itérations seront gérées dans une boucle, il est conseillé de vérifier que la fin du programme n'a pas été demandée avec la fonction IsStopped().

Exemple :

```
//--- Calcule les séries de Fibonacci
   int counterFibonacci=15;
   int i=0,first=0,second=1;
   int currentFibonacciNumber;
   do
     {
      currentFibonacciNumber=first+second;
      Print("i = ",i,"  currentFibonacciNumber = ",currentFibonacciNumber);
      first=second;
      second=currentFibonacciNumber;
      i++; // sans cet opérateur, une boucle infinie serait générée !
     }
   while(i<counterFibonacci && !IsStopped());
```

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur Break

L'opérateur break termine l'exécution de l'opérateur externe imbriqué switch, while, do-while ou for le plus proche. Le contrôle est passé à l'opérateur qui suit celui qui est terminé. L'un des buts de cet opérateur est de terminer l'exécution en boucle lorsqu'une certaine valeur est affectée à une variable.

Exemple :

```
//--- recherche du premier élément égal à zéro
for(i=0;i<array_size;i++)
  if(array[i]==0)
    break;
```

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur Continue

L'opérateur continue passe le contrôle au début de l'itération suivante de l'opérateur de boucle while, do-while ou for. Le but de cet opérateur est l'opposé de celui de l'opérateur break.

Exemple :

```
//--- Somme de tous les éléments différents de zéro
int func(int array[])
  {
   int array_size=ArraySize(array);
   int sum=0;
   for(int i=0;i<array_size; i++)
     {
      if(a[i]==0) continue;
      sum+=a[i];
     }
   return(sum);
  }
```

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur de Création d'Objet new

L'opérateur new crée automatiquement un objet de la taille désirée, appelle le constructeur de l'objet et retourne un descripteur sur l'objet créé. En cas d'échec, l'opérateur retourne un descripteur null qui peut être comparé avec la constante NULL.

L'opérateur new ne peut être appliqué qu'aux objets de classe objects. Il ne peut pas être appliqué aux structures.

L'opérateur ne devrait pas être utilisé pour créer des tableaux d'objets. Pour cela, utilisez la fonction ArrayResize().

Exemple :

```
//+------------------------------------------------------------------+
//| Création de la figure                                            |
//+------------------------------------------------------------------+
void CTetrisField::NewShape()
  {
   m_ypos=HORZ_BORDER;
//--- crée l'une des 7 formes possibles de façon aléatoire
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- dessin
   if(m_shape!=NULL)
     {
      //--- pré-configuration
      m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
      m_shape.SetYPos(m_ypos);
      m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
      //--- dessin
      m_shape.Draw();
     }
//---
  }
```

Il est à noter que le descripteur de l'objet n'est pas un pointeur sur une adresse mémoire.

Un objet créé avec l'opérateur new doit être supprimé de façon explicite avec l'opérateur delete.

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Opérateur de Suppression d'Objet delete

L'opérateur delete supprime un objet créé avec l'opérateur new, appelle le destructeur de la la classe correspondante et libère la mémoire occupée par l'objet. Un descripteur réel d'un objet existant est utilisé comme opérande. Une fois que l'opération de suppression est exécutée, le descripteur de l'objet devient invalide.

Exemple :

```
//--- supprime la figure
delete m_shape;
m_shape=NULL;
//--- crée une nouvelle figure
NewShape();
```

Voir aussi

Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Fonctions

Chaque tâche peut être divisée en sous-tâches, dont chacune peut être soit directement représentée sous la forme de code, ou divisée en sous-tâches plus petites. Cette méthode est appelée *raffinement par étapes*. Les fonctions sont utilisées pour écrire le code des sous-tâches à résoudre. Le code décrivant ce qu'une fonction effectue est appelé la *définition de la fonction* :

```
en-tête_de_la_fonction
  {
   instructions
  }
```

Tout ce qui se trouve avant la première accolade est l'*en-tête* de la définition de fonction, et tout ce qui est entre les accolades correspond au *corps* de la définition de la fonction. L'en-tête de la fonction inclut une description du type de la valeur de retour, le nom de la fonction (identifiant) et les paramètres formels. Le nombre de paramètres passés à la fonction est limité et ne peut pas être supérieur à 64.

La fonction peut être appelée depuis d'autres parties du programmes et autant de fois que nécessaire. En fait, le type de retour, l'identifiant de la fonction et les types des paramètres constituent le *prototype de la fonction*.

Le prototype de la fonction est la déclaration de la fonction, mais pas sa définition. En raison de la déclaration explicite du type de retour et de la liste des types des arguments, la vérification stricte du type et le casting de type implicite sont possibles pendant les appels à la fonction. Très souvent, les déclarations de fonctions sont utilisées dans les classes pour améliorer la lisibilité du code.

La définition de la fonction doit correspondre exactement à sa déclaration. Chaque fonction déclarée doit être définie.

Exemple :

```
double                        // type de la valeur de retour
linfunc (double a, double b)  // nom de la fonction et liste de paramètres
  {
                              // opérateur composite
   return (a + b);            // valeur de retour
  }
```

L'opérateur return peut retourner la valeur d'une expression située dans cet opérateur. Si nécessaire, la valeur de l'expression est convertie dans le type du résultat de la fonction. Ce qui peut être retourné : types simples, structures simples, pointeurs d'objets. Avec l'opérateur *return*, il n'est pas possible de retourner des tableaux, des objets classe, des variables de type structure composée.

Une fonction qui ne retourne aucune valeur doit être décrite comme retournant le type void.

Exemple :

```
void errmesg(string s)
  {
   Print("erreur : "+s);
  }
```

© 2000-2024, MetaQuotes Ltd.

Les paramètres passés à la fonction peuvent avoir des valeurs par défaut, qui sont définies par des constantes du même type.

Exemple :

```
int somefunc(double a,
             double d=0.0001,
             int n=5,
             bool b=true,
             string s="string passée")
  {
   Print("Paramètre requis a = ",a);
   Print("Passe les paramètres suivants : d = ",d," n = ",n," b = ",b," s = ",s);
   return(0);
  }
```

Si l'un des paramètres a une valeur par défaut, tous les paramètres suivants doivent également avoir des valeurs par défaut.

Exemple de déclaration incorrecte :

```
int somefunc(double a,
             double d=0.0001,    // valeur par défaut déclarée: 0.0001
             int n,              // la valeur par défaut n'est pas spécifiée !
             bool b,             // la valeur par défaut n'est pas spécifiée !
             string s="string passée")
  {
  }
```

Voir aussi

Surcharge, Fonctions Virtuelles, Polymorphisme

© 2000-2024, MetaQuotes Ltd.

Appel de Fonction

Si un nom qui n'a pas encore été utilisé auparavant apparaît dans l'expression et est suivi par une parenthèse ouvrante, il est contextuellement considéré comme étant le nom d'une fonction.

```
nom_de_fonction (x1, x2,..., xn)
```

Les arguments (paramètres formels) sont passés par valeur, c'est à dire que chaque expression x1,..., xn est calculée et la valeur est passée à la fonction. L'ordre de calcul des expressions et l'ordre de chargement des valeurs n'est pas garanti. A l'exécution, le système vérifie le nombre et le type des arguments passés à la fonction. Cette façon de s'adresser à la fonction est appelée un appel par valeur.

L'appel à la fonction est une expression, dont la valeur est la valeur retournée par la fonction. Le type de fonction décrit ci-dessus doit correspondre au type de la valeur de retour. La fonction peut être déclarée ou décrite à n'importe quel endroit du programme dans la portée globale, c'est à dire en dehors des autres fonctions. La fonction ne peut pas être déclarée ou décrite à l'intérieur d'une autre fonction.

Exemples :

```
int start()
  {
   double some_array[4]={0.3, 1.4, 2.5, 3.6};
   double a=linfunc(some_array, 10.5, 8);
   //...
  }
double linfunc(double x[], double a, double b)
  {
   return (a*x[0] + b);
  }
```

A l'appel d'une fonction avec les paramètres par défaut, la liste des paramètres à passer peut être limitée, mais pas avant le premier paramètre ayant une valeur par défaut.

Exemples :

```
void somefunc(double init,
              double sec=0.0001, // définit les valeurs par defaut
              int level=10);
//...
somefunc();                      // Appel incorrect. Le premier paramètre doit être p
somefunc(3.14);                  // Appel correct
somefunc(3.14,0.0002);           // Appel correct
somefunc(3.14,0.0002,10);        // Appel correct
```

Lors de l'appel à une fonction, il n'est pas possible d'ignorer les paramètres, même ceux ayant des valeurs par défaut :

```
somefunc(3.14, , 10);           // Appel incorrect -> le deuxième paramètre a été oub
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Surcharge, Fonctions Virtuelles, Polymorphisme

© 2000-2024, MetaQuotes Ltd.

Passer des Paramètres

Il existe deux méthodes grâce auxquelles le langage machine peut passer des arguments à un sous-programme (fonction). La première méthode est d'envoyer un paramètre par valeur. Cette méthode copie la valeur de l'argument dans un paramètre formel de fonction. Tout changement dans ce paramètre dans la fonction n'a donc aucune influence sur l'argument correspondant de l'appel.

```
//+------------------------------------------------------------------+
//| Passer les paramètres par valeur                                 |
//+------------------------------------------------------------------+
double FirstMethod(int i,int j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a et b avant l'appel :",a," ",b);
   double d=FirstMethod(a,b);
   Print("a et b après l'appel :",a," ",b);
  }
//--- Résultat de l'exécution du script
//  a et b avant l'appel : 14 8
//  a et b après l'appel : 14 8
```

La deuxième méthode est de passer par référence. Dans c cas, la référence à un paramètre (et pas sa valeur) est passée au paramètre d'une fonction. Dans la fonction, elle est utilisée pour se référer au paramètre actuel spécifié dans l'appel. Cela signifie qu'un changement du paramètre affectera l'argument utilisé pour appeler la fonction.

```
//+------------------------------------------------------------------+
//| Passer des paramètres par référence                              |
//+------------------------------------------------------------------+
double SecondMethod(int &i,int &j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
```

© 2000-2024, MetaQuotes Ltd.

```
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a et b avant l'appel :",a," ",b);
   double d=SecondMethod(a,b);
   Print("a et b après l'appel :",a," ",b);
  }
//+------------------------------------------------------------------+
//--- résultat de l'exécution du script
//  a et b avant l'appel : 14 8
//  a et b après l'appel : 28 4
```

MQL5 utilise les deux méthodes, avec une exception : les tableaux, les variables de type structure et les objets de classe sont toujours passés par référence. Pour interdire de changer les paramètres courants (les arguments passés à l'appel de la fonction), utilisez le spécificateur d'accès const. En cas de tentative de changer le contenu d'une variable déclarée avec le spécificateur *const* , le compilateur génèrera une erreur.

Note

Il est à noter que les paramètres sont passés à une fonction dans l'ordre inverse, c'est à dire que le dernier paramètre est d'abord calculé et passé, ensuite l'avant dernier, etc. le dernier paramètre calculé et passé est le premier situé après la parenthèse ouvrante.

Exemple :

```
void OnStart()
  {
//---
   int a[]={0,1,2};
   int i=0;

   func(a[i],a[i++],"Premier appel (i = "+string(i)+")");
   func(a[i++],a[i],"Deuxième appel (i = "+string(i)+")");
// Résultat :
// Premier appel (i = 0) : par1 = 1     par2 = 0
// Deuxième appel (i = 1) : par1 = 1     par2 = 1

  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void func(int par1,int par2,string comment)
```

© 2000-2024, MetaQuotes Ltd.

```
{
 Print(comment,": par1 = ",par1,"    par2 = ",par2);
}
```

Dans le premier appel (voir l'exemple ci-dessus), la variable ***i*** est d'abord utilisée dans la concaténation des chaînes de caractères :

```
"Premier appel (i = "+string(i)+")"
```

Ici, sa valeur ne change pas. La variable ***i*** est ensuite utilisée dans le calcul de l'élément ***a[i++]*** du tableau, c'est à dire lors de l'accès à l'élément du tableau avec l'indice i, la ***i*** variable est incrémentée. Et seulement après, le premier paramètre avec la valeur changée de la ***i*** variable est calculé.

Dans le deuxième appel, la même valeur de i (calculée dans la première phase de l'appel de la fonction) est utilisée lors du calcul des trois paramètres. La variable ***i*** n'est encore changée qu'après le calcul du premier paramètre.

Voir aussi

Portée de Visibilité et Durée de Vie des Variables, Surcharge, Fonctions Virtuelles, Polymorphisme

© 2000-2024, MetaQuotes Ltd.

Surcharge de Fonction

Le nom de la fonction tend normalement à refléter le but principal de la fonction. La règle générale est que les programmes qui sont lisibles contiennent différents identifiants bien sélectionnés. Différentes fonctions sont parfois utilisées pour les mêmes buts. Considérons par exemple une fonction calculant la valeur moyenne d'un tableau de nombres de précision double et la même fonction, mais opérant sur un tableau d'entiers. Il est pratique d'appeler ces deux fonctions AverageFromArray :

```
//+------------------------------------------------------------------+
//| Calcul d'une moyenne d'un tableau de types doubles               |
//+------------------------------------------------------------------+
double AverageFromArray(const double & array[],int size)
  {
   if(size<=0) return 0.0;
   double sum=0.0;
   double aver;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];     // Somme des double
     }
   aver=sum/size;        // Divise la somme par le nombre
//---
   Print("Calcul de la moyenne d'un tableau de doubles");
   return aver;
  }
//+------------------------------------------------------------------+
//| Calcul d'une moyenne d'un tableau de types entiers               |
//+------------------------------------------------------------------+
double AverageFromArray(const int & array[],int size)
  {
   if(size<=0) return 0.0;
   double aver=0.0;
   int sum=0;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];      // Somme des entiers
     }
   aver=(double)sum/size;// Division de la somme par la taille
//---
   Print("Calcul de la moyenne d'un tableau d'entiers");
   return aver;
  }
```

Chaque fonction émet un message via la fonction Print() ;

```
   Print("Calcul de la moyenne d'un tableau d'entiers");
```

© 2000-2024, MetaQuotes Ltd.

Le compilateur sélectionne la fonction nécessaire selon le type des arguments et leur quantité. La règle selon laquelle le choix est effectué est appelée l'*algorithme de correspondance des signatures*. La signature est la liste de types utilisés dans la déclaration de la fonction.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int    a[5]={1,2,3,4,5};
   double b[5]={1.1,2.2,3.3,4.4,5.5};
   double int_aver=AverageFromArray(a,5);
   double double_aver=AverageFromArray(b,5);
   Print("int_aver = ",int_aver,"   double_aver = ",double_aver);
  }
//--- Résultat du script
// Calcule la moyenne d'un tableau de types int
// Calcule la moyenne d'un tableau de types doubles
// int_aver= 3.00000000    double_aver= 3.30000000
```

La surcharge de fonction est le procédé de création de plusieurs fonctions avec le même nom, mais avec des paramètres différents. Cela signifie que dans les variantes surchargées d'une fonction, le nombre d'arguments et/ou leurs types doivent être différents. Une version spécifique de la fonction est sélectionnée en fonction de la correspondance de la liste des arguments lors de l'appel de la fonction avec la liste des paramètres dans la déclaration de la fonction.

Lorsqu'une fonction surchargée est appelée, le compilateur doit avoir un algorithme pour sélectionner la fonction appropriée. L'algorithme qui effectue ce choix dépend des castings des types présents. La meilleure correspondance doit être unique. Une fonction surchargée doit être la meilleure correspondance parmi toutes les variantes pour au moins un argument. En même temps, tous les autres arguments doivent également correspondre, mais pas de façon moindre que les autres variantes.

Un algorithme de correspondance pour chaque argument est présenté ci-dessous.

Algorithme de Choix d'une Fonction Surchargée

1. Utilise la correspondante stricte (si possible).
2. Essaye l'augmentation du type standard.
3. Essaye le casting de type standard.

L'augmentation du type standard est meilleure que toutes les autres conversions standards. L'augmentation est la conversion de float vers double, de bool, char, short ou enum vers int. Le casting de type des tableaux de types integer similaires appartient également au casting de type. Les types similaires sont : bool, char, uchar, puisque les trois types sont des entiers sur un seul octet ; les entiers sur deux octets short et ushort ; les enties sur 4 octets int, uint et color ; long, ulong et datetime.

© 2000-2024, MetaQuotes Ltd.

La correspondance stricte est bien sûr la meilleure. Pour aboutir à cette consistence, le casting de type peut être utilisé. Le compilateur ne peut pas gérer des situations ambigües. Vous ne devriez donc pas vous fier aux différences subtiles des types et aux conversions implicites qui rendent les fonctions surchargées non claires.

En cas de doute, utilisez une conversion explicite pour assurer une correspondance stricte.

Des exemples de fonctions surchargées en MQL5 sont visibles dans l'exemple des fonctions ArrayInitialize().

Les règles de surcharge des fonctions s'appliquent aux méthodes de classes surchargées.

La surcharge des fonctions du système est autorisée mais il est à noter que le compilateur est capable de sélectionner la fonction nécessaire de fonction précise. Par exemple, nous pouvons surcharger la fonction système MathMax() de 4 façons différentes, mais dont seulement deux variantes sont correctes.

Exemple :

```
// 1. la surcharge est autorisée - la différence avec la fonction MathMax() intégrée e
double MathMax(double a,double b,double c);

// 2. la surcharge n'est pas autorisée !
// le nombre de paramètres est différent, mais le dernier a une valeur par défaut
// cela mène à la dissimulation de la fonction système lors de l'appel, ce qui n'est p
double MathMax(double a,double b,double c=DBL_MIN);

// 3. la surcharge est autorisée - surcharge normale par le type des paramètres a et b
double MathMax(int a,int b);

// 4. la surcharge n'est pas autorisée !
// le nombre et le type des paramètres sont les mêmes que dans la fonction d'origine c
int MathMax(double a,double b);
```

Voir aussi

Surcharge, Fonctions Virtuelles, Polymorphisme

© 2000-2024, MetaQuotes Ltd.

Surcharge d'Opération

Pour faciliter la lecture et l'écriture du code, la surcharge de certaines opérations est autorisée. Un opérateur surchargé est écrit avec le mot-clé operator. Les opérateurs suivants peuvent être surchargés :

- binaire +,-,/,*,%,<<,>>,==,!=,<,>,<=,>=,=,+=,-=,/=,*=,%=,&=,|=,^=,<<=,>>=,&&,||,&,|,^
- unaire +,-,++,--,!,~
- opérateur d'assignation =
- opérateur d'indexation []

La surcharge d'opération permet d'utiliser la notation des opérations (écrite sous la forme d'expressions simples) pour les objets complexes - structures et classes. L'écriture d'expressions en utilisant des opérations surchargées simplifie la lisibilité du code source car la complexité de l'implémentation est cachée.

Par exemple, considérons les nombres complexes qui consistent en une partie réelle et une partie imaginaire. Ils sont communément utilisés en mathématiques. Le langage MQL5 n'a pas de type de données pour représenter les nombres complexes, mais il est possible de créer un nouveau type de données sous la forme d'une structure ou classe. Déclaration de la structure complexe et définition de 4 méthodes qui implémentent quatre opérations arithmétiques :

```
//+------------------------------------------------------------------+
//| Structure pour utiliser les nombres complexes                    |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // Partie réelle
   double            im; // Partie imaginaire
   //--- Constructeurs
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- Opérations arithmétiques
   complex           Add(const complex &l,const complex &r) const;  // Addition
   complex           Sub(const complex &l,const complex &r) const;  // Soustraction
   complex           Mul(const complex &l,const complex &r) const;  // Multiplication
   complex           Div(const complex &l,const complex &r) const;  // Division
  };
```

Nous pouvons maintenant déclarer dans notre code des variables représentant des nombres complexes et travailler avec eux.

Par exemple :

```
void OnStart()
  {
//--- Déclare et initialise des variables de type complexe
   complex a(2,4),b(-4,-2);
```

© 2000-2024, MetaQuotes Ltd.

```
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- Addition de deux nombres
   complex z;
   z=a.Add(a,b);
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- Multiplication de deux nombres
   z=a.Mul(a,b);
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- Division de deux nombres
   z=a.Div(a,b);
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
```

Mais il serait plus pratique d'utiliser les opérateurs habituels "+", "-", "*" et "/" pour les opérations arithmétiques ordinaires avec des nombres complexes.

Le mot-clé operator est utilisé pour définir une fonction membre qui effectue une conversion de type. Les opérations unaires et binaires pour des variables de type objet de classe peuvent être surchargées comme des fonctions membres non statiques. Ils agissent implicitement sur l'objet de classe.

La plupart des opérations binaires peuvent être surchargées comme des fonctions classiques qui prennent un ou deux arguments comme variable de classe ou un pointeur vers un objet de cette classe. Pour notre type complexe, la surcharge dans la déclaration ressemblera à :

```
   //--- Opérateurs
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
```

L'exemple complet du script :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Déclare et initialise des variables de type complexe
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
   //a.re=5;
   //a.im=1;
   //b.re=-1;
   //b.im=-5;
//--- Addition de deux nombres
   complex z=a+b;
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- Multiplication de deux nombres

   z=a*b;
```

© 2000-2024, MetaQuotes Ltd.

```
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- Division de deux nombres
   z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
//+------------------------------------------------------------------+
//| Structure pour utiliser les nombres complexes                    |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // Partie réelle
   double            im; // Partie imaginaire
   //--- Constructeurs
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- Opérations arithmétiques
   complex           Add(const complex &l,const complex &r) const;  // Addition
   complex           Sub(const complex &l,const complex &r) const;  // Soustraction
   complex           Mul(const complex &l,const complex &r) const;  // Multiplication
   complex           Div(const complex &l,const complex &r) const;  // Division
   //--- Opérateurs binaires
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
  };
//+------------------------------------------------------------------+
//| Addition                                                         |
//+------------------------------------------------------------------+
complex complex::Add(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re+r.re;
   res.im=l.im+r.im;
//--- Résultat
   return res;
  }
//+------------------------------------------------------------------+
//| Soustraction                                                     |
//+------------------------------------------------------------------+
complex complex::Sub(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re-r.re;
```

© 2000-2024, MetaQuotes Ltd.

```
   res.im=l.im-r.im;
//--- Résultat
   return res;
  }
//+------------------------------------------------------------------+
//| Multiplication                                                   |
//+------------------------------------------------------------------+
complex complex::Mul(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re*r.re-l.im*r.im;
   res.im=l.re*r.im+l.im*r.re;
//--- Résultat
   return res;
  }
//+------------------------------------------------------------------+
//| Division                                                         |
//+------------------------------------------------------------------+
complex complex::Div(const complex &l,const complex &r) const
  {
//--- Nombre complexe vide
   complex res(EMPTY_VALUE,EMPTY_VALUE);
//--- Vérifie si zéro
   if(r.re==0 && r.im==0)
     {
      Print(__FUNCTION__+" : le nombre est zéro");
      return(res);
     }
//--- Variables auxiliaires
   double e;
   double f;
//--- Sélection de la variante de calcul
   if(MathAbs(r.im)<MathAbs(r.re))
     {
      e = r.im/r.re;
      f = r.re+r.im*e;
      res.re=(l.re+l.im*e)/f;
      res.im=(l.im-l.re*e)/f;
     }
   else
     {
      e = r.re/r.im;
      f = r.im+r.re*e;
      res.re=(l.im+l.re*e)/f;
      res.im=(-l.re+l.im*e)/f;
     }
//--- Résultat
   return res;
```

© 2000-2024, MetaQuotes Ltd.

```
  }
```

La plupart des opérations unaires pour les classes peuvent être surchargées comme des fonctions ordinaires qui acceptent un seul argument de type objet ou un pointeur dessus. Ajout de la surcharge des opérations unaires "-" et "!".

```
//+------------------------------------------------------------------+
//| Structure pour utiliser les nombres complexes                    |
//+------------------------------------------------------------------+
struct complex
  {
   double            re;       // Partie réelle
   double            im;       // Partie imaginaire
...
   //--- Opérateurs unaires
   complex operator-()  const; // Opérateur Moins unaire
   bool    operator!()  const; // Négation
  };
...
//+------------------------------------------------------------------+
//| Surcharge de l'opérateur unaire "Moins"                          |
//+------------------------------------------------------------------+
complex complex::operator-() const
  {
   complex res;
//---
   res.re=-re;
   res.im=-im;
//--- Résultat
   return res;
  }
//+------------------------------------------------------------------+
//| Surcharge de l'opérateur "négation logique"                      |
//+------------------------------------------------------------------+
bool complex::operator!() const
  {
//--- Les parties réelle et imaginaire du nombre complexe sont-elles égales à zéro ?
   return (re!=0 && im!=0);
  }
```

Nous pouvons maintenant vérifier la valeur d'un nombre complexe pour zéro et récupérer une valeur négative :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Déclare et initialise des variables de type complexe
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- Division des deux nombres
   complex z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//--- Un nombre complexe est égal à zéro par défaut (dans le constructeur par défaut
   complex zero;
   Print("!zero=",!zero);
//--- Assigne une valeur négative
   zero=-z;
   PrintFormat("z=%.2f+i*%.2f,  zero=%.2f+i*%.2f",z.re,z.im, zero.re,zero.im);
   PrintFormat("-zero=%.2f+i*%.2f",-zero.re,-zero.im);
//--- Vérifie si zéro encore une fois
   Print("!zero=",!zero);
//---
  }
```

Notez qu'il n'est pas besoin de surcharger l'opérateur d'assignement "=", puisque les structures de types simples peuvent être copiées directement l'une dans l'autre. Nous pouvons donc maintenant écrire du code pour les calculs impliquant des nombres complexes d'une manière habituelle.

La surcharge de l'opérateur d'indexation permet d'obtenir les valeurs des tableaux inclus dans un objet, d'une façon simple et familière, et permet également de contribuer à une meilleure lisibilité du code source. Par exemple, nous devons fournir l'accès à un symbole dans la chaîne de caractères à la position spécifiée. Une chaîne de caractères en MQL5 est représentée par le type string, qui n'est pas un tableau de symboles, mais grâce à un opérateur d'indexation surchargé, nous pouvons fournir une utilisation simple et transparente dans la classe CString générée :

```
//+------------------------------------------------------------------+
//| Classe d'accès aux symboles d'une string comme dans un tableau   |
//+------------------------------------------------------------------+
class CString
  {
   string            m_string;

public:
                     CString(string str=NULL):m_string(str) { }
   ushort operator[] (int x) { return(StringGetCharacter(m_string,x)); }
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Un tableau pour prendre les symboles d'une chaîne de caractères
   int     x[]={ 19,4,18,19,27,14,15,4,17,0,19,14,17,27,26,28,27,5,14,
                 17,27,2,11,0,18,18,27,29,30,19,17,8,13,6 };
   CString str("abcdefghijklmnopqrstuvwxyz[ ]CS");
   string  res;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Construit une phrase en utilisant les symboles de la variable str
   for(int i=0,n=ArraySize(x);i<n;i++)
     {
      res+=ShortToString(str[x[i]]);
     }
//--- Affiche le résultat
   Print(res);
  }
```

Un autre exemple de surcharge de l'opération d'indexation est les opérations sur les matrices. La matrice représente un tableau dynamique à 2 dimensions, dont la taille n'est pas définie à l'avance. Vous ne pouvez donc pas déclarer un tableau de la forme array[][] sans spécifier la taille de la deuxième dimension, et passer ensuite ce tableau comme paramètre. Une solution possible est une classe spéciale CMatrix, contenant un tableau d'objets de la classe CRow.

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Addition et multiplication de matrices
   CMatrix A(3),B(3),C();
//--- Prépare un tableau pour les lignes
   double a1[3]={1,2,3}, a2[3]={2,3,1}, a3[3]={3,1,2};
   double b1[3]={3,2,1}, b2[3]={1,3,2}, b3[3]={2,1,3};
//--- Remplit les matrices
   A[0]=a1; A[1]=a2; A[2]=a3;
   B[0]=b1; B[1]=b2; B[2]=b3;
//--- Ecrit les matrices dans le journal des Experts
   Print("---- Eléments de la matrice A");
   Print(A.String());
   Print("---- Eléments de la matrice B");
   Print(B.String());

//--- Addition des matrices
   Print("---- Addition des matrices A et B");
   C=A+B;
//--- Affiche la représentation sous forme d'une chaîne de caractères
   Print(C.String());

//--- Multiplication des matrices
   Print("---- Multiplication des matrices A et B");
   C=A*B;
   Print(C.String());

//--- Affiche comment les valeurs sont récupérées comme dans des tableaux dynamiques r
   Print("Affiche les valeurs des éléments de la matrice C");
//--- Parcours les rangées de la matrice - des objets CRow - dans une boucle
   for(int i=0;i<3;i++)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       string com="| ";
       //--- Parcours les rangées de la matrice pour la valeur
       for(int j=0;j<3;j++)
         {
          //--- Récupère l'élément de la matrice par son numéro de ligne et de colonne
          double element=C[i][j];// [i] - Accès à l'élément CRow du tableau m_rows[] ,
                                 // [j] - Opérateur surchargé pour l'indexation dans CR
          com=com+StringFormat("a(%d,%d)=%G ; ",i,j,element);
         }
       com+="|";
       //--- Affiche les valeurs de la rangée
       Print(com);
      }
   }
//+------------------------------------------------------------------+
//| Classe "Row"                                                     |
//+------------------------------------------------------------------+
class CRow
   {
private:
    double            m_array[];
public:
    //--- Constructeurs et un destructeur
                      CRow(void)          { ArrayResize(m_array,0);    }
                      CRow(const CRow &r) { this=r;                    }
                      CRow(const double &array[]);
                     ~CRow(void){};
    //--- Nombre d'éléments dans la rangée
    int               Size(void) const    { return(ArraySize(m_array));}
    //--- Retourne une chaîne de caractères avec les valeurs
    string            String(void) const;
    //--- Opérateur d'indexation
    double            operator[](int i) const  { return(m_array[i]);   }
    //--- Opérateurs d'assignement
    void              operator=(const double  &array[]); // An array
    void              operator=(const CRow & r);         // Another CRow object
    double            operator*(const CRow &o);          // CRow object for multiplicat
   };
//+------------------------------------------------------------------+
//| Constructeur pour initialiser une rangée à partir d'un tableau   |
//+------------------------------------------------------------------+
void  CRow::CRow(const double &array[])
   {
    int size=ArraySize(array);
//--- Si le tableau n'est pas vide
    if(size>0)
      {
       ArrayResize(m_array,size);
```

```
      //--- Remplissage avec les valeurs
      for(int i=0;i<size;i++)
         m_array[i]=array[i];
     }
//---
  }
//+------------------------------------------------------------------+
//| Opération d'assignement pour le tableau                          |
//+------------------------------------------------------------------+
void CRow::operator=(const double &array[])
  {
   int size=ArraySize(array);
   if(size==0) return;
//--- Remplissage du tableau avec les valeurs
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| Opération d'assignement pour CRow                                |
//+------------------------------------------------------------------+
void CRow::operator=(const CRow  &r)
  {
   int size=r.Size();
   if(size==0) return;
//--- Remplissage du tableau avec les valeurs
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=r[i];
//---
  }
//+------------------------------------------------------------------+
//| Opérateur de multiplication par une autre rangée                 |
//+------------------------------------------------------------------+
double CRow::operator*(const CRow &o)
  {
   double res=0;
//--- Vérifications
   int size=Size();
   if(size!=o.Size() || size==0)
     {
      Print(__FUNCSIG__," : Echec de la multiplication de deux matrices, leurs tailles
      return(res);
     }
//--- Multiplie les tableaux élément par élément et ajoute les produits
   for(int i=0;i<size;i++)
      res+=m_array[i]*o[i];
//--- Résultat
   return(res);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Retourne une une chaîne de caractères formatée                   |
//+------------------------------------------------------------------+
string CRow::String(void) const
  {
   string out="";
//--- Si la taille du tableau est supérieure à zéro
   int size=ArraySize(m_array);
//--- On ne travaille qu'avec un tableau contenant des éléments
   if(size>0)
     {
      out="{";
      for(int i=0;i<size;i++)
        {
         //--- Collecte les valeurs dans une chaîne de caractères
         out+=StringFormat(" %G;",m_array[i]);
        }
      out+=" }";
     }
//--- Résultat
   return(out);
  }
//+------------------------------------------------------------------+
//| Classe "Matrix"                                                  |
//+------------------------------------------------------------------+
class CMatrix
  {
private:
   CRow              m_rows[];

public:
   //--- Constructeurs et un destructeur
                     CMatrix(void);
                     CMatrix(int rows)  { ArrayResize(m_rows,rows);            }
                    ~CMatrix(void){};
   //--- Récupère les tailles des matrices
   int               Rows()       const { return(ArraySize(m_rows));           }
   int               Cols()       const { return(Rows()>0? m_rows[0].Size():0); }
   //--- Retourne la valeur de la colonne sous la forme d'une rangée CRow
   CRow              GetColumnAsRow(const int col_index) const;
   //--- Retourne une chaîne de caractères avec les valeurs de la matrice
   string            String(void) const;
   //--- L'opérateur d'indexation retourne une chaîne de caractères par son numéro
   CRow *operator[](int i) const        { return(GetPointer(m_rows[i]));        }
   //--- Opérateur d'addition
   CMatrix           operator+(const CMatrix &m);
   //--- Opérateur de multiplication
   CMatrix           operator*(const CMatrix &m);
   //--- Opérateur d'assignation
```

© 2000-2024, MetaQuotes Ltd.

```
   CMatrix          *operator=(const CMatrix &m);
  };
//+------------------------------------------------------------------+
//| Constructeur par défaut, crée un tableau de rangées de taille 0  |
//+------------------------------------------------------------------+
CMatrix::CMatrix(void)
  {
//--- La rangée numéro 0 dans la matrice
   ArrayResize(m_rows,0);
//---
  }
//+------------------------------------------------------------------+
//| Retourne la valeur de la colonne sous la forme de CRow           |
//+------------------------------------------------------------------+
CRow  CMatrix::GetColumnAsRow(const int col_index) const
  {
//--- Variable pour récupérer les valeurs de la colonne
   CRow row();
//--- Le nombre de rangées dans la matrice
   int rows=Rows();
//--- Si le nombre de rangées est supérieur à 0, exécute l'opération
   if(rows>0)
     {
      //--- Tableau pour récupérer les valeurs de la colonne ayant l'index col_index
      double array[];
      ArrayResize(array,rows);
      //--- Remplissage du tableau
      for(int i=0;i<rows;i++)
        {
         //--- Vérifie le numéro de la colonne pour la rangée i - il peut être supérie
>         if(col_index>=this[i].Size())
           {
            Print(__FUNCSIG__," : Erreur ! Numéro de colonne ",col_index,"> taille de
            break; // rangée sera un objet non initialisé
           }
         array[i]=this[i][col_index];
        }
      //--- Crée une rangée CRow basée sur les valeurs du tableau
      row=array;
     }
//--- Résultat
   return(row);
  }
//+------------------------------------------------------------------+
//| Addition de deux matrices                                        |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator+(const CMatrix &m)
  {
//--- Le nombre de rangées et de colonnes dans la matrice passée
```

```
   int cols=m.Cols();
   int rows=m.Rows();
//--- La matrice recevant les résultats de l'addition
   CMatrix res(rows);
//--- Les tailles des matrices doivent correspondre
   if(cols!=Cols() || rows!=Rows())
     {
      //--- Addition impossible
      Print(__FUNCSIG__," : Echec de l'addition des 2 matrices, leurs tailles sont di
      return(res);
     }
//--- Tableau supplémentaire
   double arr[];
   ArrayResize(arr,cols);
//--- Se rend aux rangées à ajouter
   for(int i=0;i<rows;i++)
     {
      //--- Ecrit les résultats de l'addition des chaînes de caractères des matrices c
      for(int k=0;k<cols;k++)
        {
         arr[k]=this[i][k]+m[i][k];
        }
      //--- Place le tableau dans la rangée de la matrice
      res[i]=arr;
     }
//--- retourne le résultat de l'addition des matrices
   return(res);
  }
//+------------------------------------------------------------------+
//| Multiplication de deux matrices                                  |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator*(const CMatrix &m)
  {
//--- Nombre de colonnes de la première matrics, nombre de rangées de la matrice passé
   int cols1=Cols();
   int rows2=m.Rows();
   int rows1=Rows();
   int cols2=m.Cols();
//--- La matrice recevant les résultats de la multiplication
   CMatrix res(rows1);
//--- Les matrices doivent être coordonnées
   if(cols1!=rows2)
     {
      //--- Multiplication impossible
      Print(__FUNCSIG__," : Echec de la multiplication des 2 matrices, les formats ne
            "- le nombre de colonnes du premier facteur doit être égal au nombre de ra
      return(res);
     }
//--- Tableau supplémentaire
```

© 2000-2024, MetaQuotes Ltd.

```
   double arr[];
   ArrayResize(arr,cols1);
//--- Remplit les rangées de la matrice de multiplication
   for(int i=0;i<rows1;i++)// Boucle sur les rangées
     {
      //--- Réinitialise le tableau de destination
      ArrayInitialize(arr,0);
      //--- Boucle sur les éléments de la rangée
      for(int k=0;k<cols1;k++)
        {
         //--- Prend la valeur de la colonne k de la matrice m dans la rangée CRow
         CRow column=m.GetColumnAsRow(k);
         //--- Multiplie deux rangées et écrit le résultat de la multiplication scala
         arr[k]=this[i]*column;
        }
      //--- place le tableau arr[] à la i-ème rangée de la matrice
      res[i]=arr;
     }
//--- Retourne le produit des deux matrices
   return(res);
  }
//+------------------------------------------------------------------+
//| Opération d'assignation                                          |
//+------------------------------------------------------------------+
CMatrix *CMatrix::operator=(const CMatrix &m)
  {
//--- Trouve et définit le nombre de rangées
   int rows=m.Rows();
   ArrayResize(m_rows,rows);
//--- Remplit les rangées avec les valeurs des rangées de la matrice passée en argumen
   for(int i=0;i<rows;i++) this[i]=m[i];
//---
   return(GetPointer(this));
  }
//+------------------------------------------------------------------+
//| Représentation textuelle de la matrice                           |
//+------------------------------------------------------------------+
string CMatrix::String(void) const
  {
   string out="";
   int rows=Rows();
//--- Forme la chaîne de caractères
   for(int i=0;i<rows;i++)
     {
      out=out+this[i].String()+"\r\n";
     }
//--- Résultat
   return(out);
  }
```

© 2000-2024, MetaQuotes Ltd.

Voir aussi

Surcharge, Opérations Arithmétiques, Surcharge de Fonction, Règles de Précédence

© 2000-2024, MetaQuotes Ltd.

Description des Fonctions Externes

Les fonctions externes décrites dans un autre module doivent être décrites explicitement. La description inclut le type de retour, le nom de la fonction et les paramètres d'entrée avec leurs types. L'absence de description peut provoquer des erreurs au moment de la compilation, du build ou de l'exécution du programme. Lors de la description d'un objet externe, utilisez le mot-clé *#import* indiquant le module.

Exemples :

```
#import "user32.dll"
  int     MessageBoxW(int hWnd ,string szText,string szCaption,int nType);
  int     SendMessageW(int hWnd,int Msg,int wParam,int lParam);
#import "lib.ex5"
  double  round(double value);
#import
```

Grâce au mot-clé import, il est facile de décrire des fonctions qui sont appelées depuis une DLL externe ou une bibliothèque EX5 compilée. Les bibliothèques EX5 sont des fichiers ex5 compilés ayant la propriété library. Seules les fonctions décrites avec le modificateur export peuvent être importées depuis les bibliothèques EX5.

Veuillez noter que les bibliothèques DLL et EX5 doivent avoir des noms différents (indépendamment du répertoire où elles sont situées) si elles sont importées ensemble. Toutes les fonctions importées ont une portée correspondant au "nom de fichier" de la bibliothèque.

Exemple :

```
#import "kernel32.dll"
   int GetLastError();
#import "lib.ex5"
   int GetLastError();
#import

class CFoo
  {
public:
   int            GetLastError() { return(12345); }
   void           func()
     {
      Print(GetLastError());           // appel de la méthode de classe
      Print(::GetLastError());         // appel de la fonction MQL5
      Print(kernel32::GetLastError()); // appel de la fonction de la DLL kernel32.dll
      Print(lib::GetLastError());      // appel de la fonction de la bibliothèque EX5
     }
  };

void OnStart()
  {
   CFoo foo;
   foo.func();
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

Voir aussi

Surcharge, Fonctions Virtuelles, Polymorphisme

© 2000-2024, MetaQuotes Ltd.

Export de Fonctions

Une fonction déclarée dans une programme mql5 avec le post-modificateur *export* peut être utilisée dans un autre programme mql5. Une telle fonction est dite exportable, et peut être appelée depuis un autre programme après la compilation.

```
int Function() export
  {
  }
```

Ce modificateur ordonne au compilateur d'ajouter la fonction dans la table des fonctions EX5 exportées par ce fichier ex5. Seules les fonctions avec ce modificateur sont accessibles ("visibles") depuis d'autres programmes mql5.

La propriété library indique au compilateur que le fichier EX5 sera une bibliothèque, et le compilateur le montrera dans l'en-tête du EX5.

Toutes les fonctions à exporter doivent être marquées avec le modificateur *export*.

Voir aussi

Surcharge, Fonctions Virtuelles, Polymorphisme

© 2000-2024, MetaQuotes Ltd.

Fonctions de Gestion d'Evènement

Le langage MQL5 permet de gérer certains évènements prédéfinis. Les fonctions pour gérer ces évènements doivent être définies dans un programme MQL5 ; le nom de la fonction, le type de retour, la composition des paramètres (s'il y en a) et leurs types doivent correspondre exactement à la description de chaque fonction de gestion des évènements.

Le gestionnaire d'évènements du terminal client identifie les fonctions gérant tel ou tel évènement par le type de la valeur de retour et des paramètres. Si d'autres paramètres ne correspondant pas aux descriptions ci-dessous sont spécifiés pour une certaine fonction, ou qu'un autre type de retour est indiqué, cette fonction ne sera pas utilisée comme gestionnaire d'évènement.

OnStart

La fonction OnStart() est le gestionnaire de l'évènement Start, qui n'est automatiquement généré **que** pour exécuter des **scripts**. Elle doit être de type void sans aucun paramètre :

```
void OnStart();
```

Pour la fonction OnStart(), le type de retour int peut être spécifié.

OnInit

La fonction OnInit() est le gestionnaire de l'évènement Init. Elle doit être de type void ou int, sans aucun paramètre :

```
void OnInit();
```

L'évènement Init est généré immédiatement après qu'un Expert Advisor ou qu'un indicateur est chargé ; cet évènement n'est pas généré pour les scripts. La fonction OnInit() est utilisé pour l'initialisation. Si OnInit() a une valeur de retour de type int, un code de retour différent de zéro signifie un échec de l'initialisation et l'évènement Deinit est généré avec le code de la raison de la désinitialisation REASON_INITFAILED.

Pour optimiser les paramètres d'entrée d'un Expert Advisor, il est recommandé d'utiliser les valeurs de l'énumération ENUM_INIT_RETCODE comme code de retour. Ces valeurs sont utilisées pour gérer le déroulé de l'optimisation, incluant la sélection des agents de test les plus appropriés. Pendant l'initialisation d'un Expert Advisor avant le démarrage du test, vous pouvez demander des informations sur la configuration et les ressources d'un agent (le nombre de coeurs, la quantité de mémoire libre, etc.) avec la fonction TerminalInfoInteger(). Selon les informations reçues, vous pouvez soit décider d'utiliser cet agent de test, ou de le rejeter pendant l'optimisation de cet Expert Advisor.

ENUM_INIT_RETCODE

Identifiant	Description
INIT_SUCCEEDED	Initialisation réussie avec succès, le test de l'Expert Advisor peut continuer. Ce code est identique à la valeur null - l'Expert Advisor a été initialisé avec succès dans le tester.
INIT_FAILED	L'initialisation a échoué ; il n'y a aucune raison de continuer le test en raison d'erreurs fatales. Par exemple, échec de création

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description
	d'un indicateur qui est nécessaire pour l'utilisation de l'Expert Advisor. Cette valeur de retour est identique à une valeur différente de zéro - l'initialisation de l'Expert Advisor dans le tester a échoué.
INIT_PARAMETERS_INCORRECT	Cette valeur indique un ensemble incorrect de paramètres d'entrée. La chaîne de caractères résultat contenant ce code de retour est surlignée en rouge dans le tableau général d'optimisation. Le test de l'Expert Advisor pour l'ensemble de paramètres donnés ne sera pas effectué, l'agent est libre de recevoir une nouvelle tâche. Lorsqu'il reçoit cette valeur, le strategy tester ne passe pas cette tâche à d'autres agents pour réessayer.
INIT_AGENT_NOT_SUITABLE	Aucune erreur ne s'est produite pendant l'initialisation, mais pour une raison quelconque, l'agent n'est pas disponible pour effectuer un test. Par exemple, pas assez de mémoire, pas de support d'OpenCL, etc. Après le retour de ce code, l'agent ne recevra plus aucune tâche jusqu'à la fin de cette optimisation.

La fonction OnInit() du type void dénote toujours une initialisation réussie.

OnDeinit

La fonction OnDeinit() est appelée pendant la désinitialisation et est le gestionnaire de l'évènement Deinit. Elle doit être déclarée comme retournant un type void et ne doit avoir qu'un seul paramètre de type const int, contenant le code de la raison de la désinitialisation. Si un type différent est déclaré, le compilateur génère un avertissement, mais la fonction ne peut pas être appelée. Pour les scripts, l'évènement Deinit n'est pas généré et la fonction OnDeinit() ne peut donc pas être utilisée dans les scripts.

```
void OnDeinit(const int reason);
```

L'évènement Deinit est généré pour les Expert Advisors et pour les indicateurs dans les cas suivants :

- avant une réinitialisation en raison du changement du symbole ou de la période du graphique auquel le programme mql5 est attaché ;
- avant une réinitialisation en raison du changement des paramètres d'entrée ;
- avant le déchargemnt du programme mql5.

OnTick

L'évènement NewTick est généré pour les **Expert Advisors uniquement** lorsqu'un nouveau tick est reçu pour le symbole auquel l'Expert Advisor est attaché. Il est inutile de définir la fonction OnTick() dans un indicateur personnalisé ou dans un script car l'évenement NewTick n'est pas généré pour eux.

© 2000-2024, MetaQuotes Ltd.

L'évènement Tick n'est généré que pour les Expert Advisors, mais cela ne signifie pas que les Expert Advisors ont besoin de la fonction OnTick(), puisque seuls les évènements NewTick sont générés pour les Expert Advisors, mais les évènements Timer, BookEvent et ChartEvent sont également générés. Elle doit être déclarée de type void, sans aucun paramètre :

```
void OnTick();
```

OnTimer

La fonction OnTimer() est appelée lorsque l'évènement Timer survient. Il n'est généré par la minuterie du système que pour les Expert Advisors et les indicateurs - il ne peut pas être utilisé dans les scripts. La fréquence de survenue de l'évènement est définie lors de l'abonnement aux notifications de cet évènement et devant être reçus par la fonction EventSetTimer().

Vous pouvez désenregistrer un certain Expert Advisor pour ne plus reçevoir les évènements de minuterie en utilisant la fonction EventKillTimer(). La fonction doit être déclarée retournant le type void, sans aucun paramètre :

```
void OnTimer();
```

Il est recommandé d'appeler la fonction EventSetTimer() dans la fonction OnInit(), et la fonction EventKillTimer() devrait être appelée dans OnDeinit().

Chaque Expert Advisor, de même que chaque indicateur, fonctionne avec sa propre minuterie et ne reçoit les évènements que de lui. Dès que le programme mql5 s'arrête, la minuterie est détruite en force, si elle a été créée mais n'a pas été désactivée par la fonction EventKillTimer().

OnTrade

La fonction est appelée lorsque l'évènement Trade survient, qui apparaît lorsque vous changer la liste des ordres placés et des positions ouvertes, l'historique des ordres et l'historique des transactions. Lorsqu'une activité de trading est effectuée (ouverture d'un ordre en attente, ouverture/fermeture d'une position, mise en place d'un stop, déclenchemet d'un ordre en attente, etc.), l'historique des ordres et des transactions et/ou la liste des positions et des ordres en cours sont changés conforméménet.

```
void OnTrade();
```

Les utilisateurs doivent implémenter de façon indépendante dans le code la vérification de l'état du compte de trading lorsqu'un évènement est reçu (si cela est requis par les conditions de la stratégie de trading). Si l'appel à la fonction OrderSend() s'est terminé avec succès et a retourné true, cela signifie que le serveur de trading a placé l'ordre dans la queue pour être exécuté et lui a assigné un numéro de ticket. Dès que le serveur a traité cet ordre, l'évènement Trade est généré. Et si un utilisateur a toujours la valeur du ticket, il peut savoir ce qu'il s'est passé sur l'ordre avec cette valeur pendant la gestion de l'évènement OnTrade().

OnTradeTransaction

Lorsque certaines actions sont effectuées sur un compte de trading, son état change. Ces actions correspondent à :

- L'envoi d'une demande de trade depuis une application MQL5 dans le terminal client en utilisant les fonctions OrderSend et OrderSendAsync et leurs exécutions ultérieures ;

© 2000-2024, MetaQuotes Ltd.

- L'envoi d'une demande de trade via l'interface graphique du terminal et son exécution ultérieure ;
- L'action des ordres en attente et des ordres stop sur le serveur ;
- L'exécution d'opérations sur le serveur de trades.

Les transactions de trading suivantes sont effectuées comme étant le résultat de ces actions :

- gestion d'une demande de trade ;
- changement d'ordres d'ouverture ;
- changement de l'historiuque des ordres ;
- changement de l'historique des transactions ;
- changement des positions.

Par exemple, lors de l'envoi d'un ordre d'achat au marché, il est géré, un ordre buy correspondant est créé pour le compte, l'ordre est alors exécuté et supprimé de la liste des ordres ouvertes, il est ensuite ajouté à l'historique des ordres, une transaction correspondante est ajoutée à l'historique et une nouvelle position est créée. Toutes ces actions sont des transactions de trading. L'arrivée d'une telle transaction dans le terminal est un évènement TradeTransaction. Il appelle le gestionnaire OnTradeTransaction

```
void  OnTradeTransaction(
   const MqlTradeTransaction&    trans,        // structure de la transaction de tradi
   const MqlTradeRequest&        request,      // structure de la demande
   const MqlTradeResult&         result        // structure du résultat
   );
```

Le gestionnaire contient trois paramètres :

- **trans** - ce paramètre prend la structure MqlTradeTransaction décrivant une transaction de trading appliquée à un compte de trading ;
- **request** - ce paramètre prend la structure MqlTradeRequest décrivant une demande de trade ;
- **result** - ce paramètre prend la structure MqlTradeResult décrivant le résultat de l'exécution de la demande de trade.

Les deux derniers paramètres **request** et **result** sont remplis par valeurs seulement pour les transactions de type TRADE_TRANSACTION_REQUEST, les données des transactions peuvent être récupérées dans le paramètrs *type* de la variable **trans**. Notez que dans ce cas, le champ *request_id* dans la variable **result** contient l'identifiant de la **demande de trade** request, après l'exécution de laquelle la transaction de trading décrite dans la variable **trans** a été effectuée. L'identifiant de la demande permet d'associer l'action effectuée (appel des fonctions OrderSend ou OrderSendAsync) avec le résultat de cette action envoyée à OnTradeTransaction().

Une demande de trade envoyée manuellement depuis le terminal ou via les fonctions OrderSend()/OrderSendAsync() peut générer plusieurs transactions consécutives sur le serveur de trades. La priorité de l'arrivée des ces transactions dans le terminal n'est pas garantie. Vous ne devez donc pas supposer qu'un groupe de transactions arrivera après un autre lors du développement de votre algorithme de trading.

© 2000-2024, MetaQuotes Ltd.

- Tous les types des transactions de trade sont décrits dans l'énumération ENUM_TRADE_TRANSACTION_TYPE.
- La structure MqlTradeTransaction décrivant une transaction de trade est remplie de différentes façons suivant le type de la transaction. Par exemple, seul le champ type (type de la transaction de trading) doit être analysé pour les transactions de type TRADE_TRANSACTION_REQUEST. Les 2ème et 3ème paramètres de la fonction OnTradeTransaction (request et result) doivent être analysés pour obtenir des données supplémentaires. Pour plus d'informations, consultez la section "Structure d'une Transaction de Trade".
- La description d'une transaction de trade ne livre pas toutes les informations disponibles concernant les ordres, les transactions et les positions (par exemple les commentaires). Il faut utiliser les fonctions OrderGet*, HistoryOrderGet*, HistoryDealGet* et PositionGet* pour obtenir plus d'informations.

Après avoir envoyé les transactions de trade pour un compte client, elles sont régulièrement placée dans la queue des transactions de trade du terminal, à partir de laquelle elles sont régulièrement envoyées vers le point d'entrée OnTradeTransaction dans l'ordre d'arrivée au terminal.

Lors du traitement des transactions de trade par un Expert Advisor avec la fonction OnTradeTransaction, le terminal continue de traiter les nouvelles transactions de trade qui continuent d'arriver. L'état d'un compte de trading peut donc changer pendant une opération OnTradeTransaction. Par exemple, alors qu'un programme MQL5 traite l'ajout d'un nouvel ordre, il peut être exécuté, supprimé de la liste des ordres ouverts et déplacé dans l'historique. Par la suite, l'application sera notifiée de ces évènements.

La longueur de la queue des transactions est de 1024 éléments. Si OnTradeTransaction traite une nouvelle transaction depuis trop longtemps, les anciennes transactions de la queue peuvent etre remplacées par les nouvelles.

- Il n'y a en général pas de rapport précis entre le nombre d'appels à OnTrade et à OnTradeTransactions. Un appel à OnTrade correspond à un ou plusieurs appels à OnTradeTransaction.
- OnTrade est appelée après les appels OnTradeTransaction correspondants.

OnTester

La fonction OnTester() est le gestionnaire de l'évènement Tester qui est automatiquement généré après que le test d'un Expert Advisor sur l'historique et sur un intervalle choisi est terminé. La fonction doit retourner un type double, et ne pas avoir de paramètres :

```
double OnTester();
```

La fonction est appelée juste avant l'appel à OnDeinit() et a le même type de valeur de retour - double. OnTester() ne peut être utilisée que lors du test d'Expert Advisors. Son intérêt principal est de calculer une certaine valeur qui est utilisée comme Critère max personnalisé dans l'optimisation génétique des paramètres d'entrée.

Dans l'optimisation génétique, le tri descendant est appliqué aux résultats en une seule génération. C'est à dire que du point de vue du critère d'optimisation, les meilleurs résultats sont ceux ayant les plus grandes valeurs (pour le Critère maximum d'optimisation, les valeurs retournées par la fonction

© 2000-2024, MetaQuotes Ltd.

OnTester sont prises en compte). Dans ce type de tri, les valeurs les moins bonnes sont positionnées à la fin et jetées ensuite et ne participet pas à la formation de la génération suivante.

OnTesterInit

La fonction OnTesterInit() est le gestionnaire de l'évènement TesterInit, qui est automatiquement généré avant le lancement de l'optimisation de l'Expert Advisor dans le strategy tester. La fonction doit être définie avec le type de retour void. Elle n'a pas de paramètres :

```
void OnTesterInit();
```

Avec le lancement de l'optimisation, un Expert Advisor ayant les fonctions OnTesterDeinit() ou OnTesterPass() est automatiquement chargé dans un graphique séparé du terminal avec le symbole et la période spécifiés dans le tester, et reçoit l'évènement TesterInit. La fonction est utilisée pour l'initialisation des Expert Advisor avant le lancement de l'optimisation pour pouvoir ensuite traiter les résultat de l'optimisation.

OnTesterPass

La fonction OnTesterPass() est le gestionnaire de l'évènement TesterPass, qui est automatiquement généré lorsqu'un nouveau cadre est reçu pendant une optimisation d'un Expert Advisor dans le strategy tester. La fonction doit être définie avec le type de retour void. Elle n'a pas de paramètres :

```
void OnTesterPass();
```

Un Expert Advisor ayant la fonction OnTesterPass() est automatiquement chargé dans un graphique séparé du terminal avec le symbole et la période spécifiés pour le test, et reçoit les évènement TesterPass lorsqu'un nouveau cadre est reçu pendant l'optimisation. La fonction est utilisée pour la gestion dynamique des résultats d'optimisation "sur le vif" sans attendre la fin de l'optimisation. Les cadres sont ajoutés avec la fonction FrameAdd(), qui peut être appelée après la fin d'une passe dans la fonction OnTester().

OnTesterDeinit

OnTesterDeinit() est le gestionnaire de l'évènement TesterDeinit, qui est automatiquement généré après la fin de l'optimisation d'un Expert Advisor dans le strategy tester. La fonction doit être définie avec le type de retour void. Elle n'a pas de paramètres :

```
void OnTesterDeinit();
```

Un Expert Advisor ayant la fonction TesterDeinit() est automatiquement chargé sur un graphique au lancement de l'optimisation, et reçoit TesterDeinit après sa fin. La fonction est utilisée pour le traitement final de tous les résultats d'optimisation.

OnBookEvent

La fonction OnBookEvent() est le gestionnaire de BookEvent. BookEvent est généré pour les Expert Advisors et les indicateurs lorsque le Depth of Market change. Elle doit être de type void et avoir un paramètre de type string :

```
void OnBookEvent (const string& symbol);
```

© 2000-2024, MetaQuotes Ltd.

Pour recevoir les évènements BookEvent pour n'importe quel symbole, nous n'avez qu'à vous inscrire pour recevoir les évènements pour ce symbole en utilisant la fonction MarketBookAdd(). Pour se désabonner des évènements BookEvent pour un symbole donnée, appelez MarketBookRelease().

Contrairement aux autres évènements, l'évènement BookEvent est diffusé. Cela signifie que si un Expert Advisor s'abonne pour recevoir les évènements BookEvent en utilisant MarketBookAdd, tous les autres Experts Advisors qui ont une fonction OnBookEvent() recevront également cet évènement. Il est donc nécessaire d'analyser le nom du symbole qui est passé à la fonction dans le paramètre *const string& symbol*.

OnChartEvent

OnChartEvent() est le gestionnaire d'un groupe d'évènements ChartEvent events:

- CHARTEVENT_KEYDOWN — évènement d'appui sur une touche lorsque la fenêtre du graphique reçoit le focus ;
- CHARTEVENT_MOUSE_MOVE — évènements de déplacement et de clics de souris (si CHART_EVENT_MOUSE_MOVE=true est défini pour le graphique) ;
- CHARTEVENT_OBJECT_CREATE — évènement de création d'un objet graphique (si CHART_EVENT_OBJECT_CREATE=true est défini pour le graphique) ;
- CHARTEVENT_OBJECT_CHANGE — évènement de changement de la propriété d'un objet via la boîte de dialogue des propriétés ;
- CHARTEVENT_OBJECT_DELETE — évènement de suppression d'un objet graphique (si CHART_EVENT_OBJECT_DELETE=true est défini pour le graphique) ;
- CHARTEVENT_CLICK — évènement de clic de souris sur le graphique ;
- CHARTEVENT_OBJECT_CLICK — évènement de clic de souris dans un objet graphique appartenant au graphique ;
- CHARTEVENT_OBJECT_DRAG — évènement de déplacement d'un objet graphique avec la souris ;
- CHARTEVENT_OBJECT_ENDEDIT — évènement de fin de l'édition du texte dans la zone de saisie d'un objet graphique LabelEdit ;
- CHARTEVENT_CHART_CHANGE — évènement de changement du graphique ;
- CHARTEVENT_CUSTOM+n — identifiant de l'évènement utilisateur, où n est dans l'intervalle de 0 à 65535.
- CHARTEVENT_CUSTOM_LAST — le dernier identifiant possible pour un évènement personnalisé (CHARTEVENT_CUSTOM +65535).

La fonction ne peut être appelée que dans les Expert Advisors et les indicateurs. La fonction doit être de type void avec 4 paramètres :

```
void OnChartEvent(const int id,          // Identifiant de l'évènement
                  const long& lparam,    // Paramètre de type long
                  const double& dparam,  // Paramètre de type double
                  const string& sparam   // Paramètre de type string
  );
```

Pour chaque type d'évènement, les paramètres d'entrée de la fonction OnChartEvent() ont des valeurs définies requises pour le traitement de cet évènement. Les évènements et les valeurs passés via ces paramètres sont listés dans le tableau ci-dessous.

© 2000-2024, MetaQuotes Ltd.

Evènement	Valeur du paramètre id	Valeur du paramètre lparam	Valeur du paramètre dparam	Valeur du paramètre sparam
Evènement d'appui sur une touche	CHARTEVENT_KEYDOWN	code de la touche pressée	Nombre de répétitions (le nombre de fois l'appui sur la touche est répété comme étant le résultat de l'utilisateur continuant à appuyer sur la touche)	La valeur textuelle du masque de bits décrivant le status des touches du clavier
Evènements souris (si la propriété CHART_EVENT_MOUSE_MOVE=true est définie pour le graphique)	CHARTEVENT_MOUSE_MOVE	la coordonnée X	la coordonnée Y	La valeur de type string du masque de bits décrivant le statut des boutons de la souris
Evènement de création d'un objet graphique (si CHART_EVENT_OBJECT_CREATE=true est défini pour le graphique)	CHARTEVENT_OBJECT_CREATE	—	—	Nom de l'objet graphique créé
Evènement de changement de propriété d'un objet via la boîte de dialogue des propriétés	CHARTEVENT_OBJECT_CHANGE	—	—	Nom de l'objet graphique modifié
Evènement de suppression d'un objet graphique (si CHART_EVENT_OBJECT_DELETE=true est défini pour le graphique)	CHARTEVENT_OBJECT_DELETE	—	—	Nom de l'objet graphique supprimé

© 2000-2024, MetaQuotes Ltd.

Evènement	Valeur du paramètre id	Valeur du paramètre lparam	Valeur du paramètre dparam	Valeur du paramètre sparam
Evènement de clic de la souris sur le graphique	CHARTEVENT_CLICK	la coordonnée X	la coordonnée Y	—
Evènement de clic de la souris sur un objet graphique appartenant au graphique	CHARTEVENT_OBJECT_CLICK	la coordonnée X	la coordonnée Y	Nom de l'objet graphique sur lequel l'évènement s'est produit
Evènement de déplacement d'un objet graphique avec la souris	CHARTEVENT_OBJECT_DRAG	—	—	Nom de l'objet graphique déplacé
Evènement de fin d'édition du texte dans la zone de saisie d'un objet graphique Champ d'Edition	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nom de l'objet graphique Champ d'Edition dans lequel l'édition du texte est terminée
Evènement de changement du graphique	CHARTEVENT_CHART_CHANGE	—	—	—
Identifiant de l'évènement utilisateur sous le numéro N	CHARTEVENT_CUSTOM+N	Valeur définie par la fonction EventChartCustom()	Valeur définie par la fonction EventChartCustom()	Valeur définie par la fonction EventChartCustom()

OnCalculate

La fonction OnCalculate() n'est appelée que pour les indicateurs personnalisés lorsqu'il est nécessaire de calculer les valeurs de l'indicateur pour l'évènement Calculate. Ceci arrive habituellement lorsqu'un nouveau tick est reçu pour le symbole pour lequel l'indicateur est calculé. Cette indicateur n'a pas besoin d'être attaché à un graphique des prix de ce symbole.

La fonction OnCalculate() doit retourner un type int. Il existe deux définitions possibles. Vous ne pouvez pas utiliser les deux versions de la fonction dans un même indicateur.

La première forme est prévue pour les indicateurs qui peuvent être calculés sur un seul buffer de données. Un exemple d'indicateur de ce type est le Custom Moving Average.

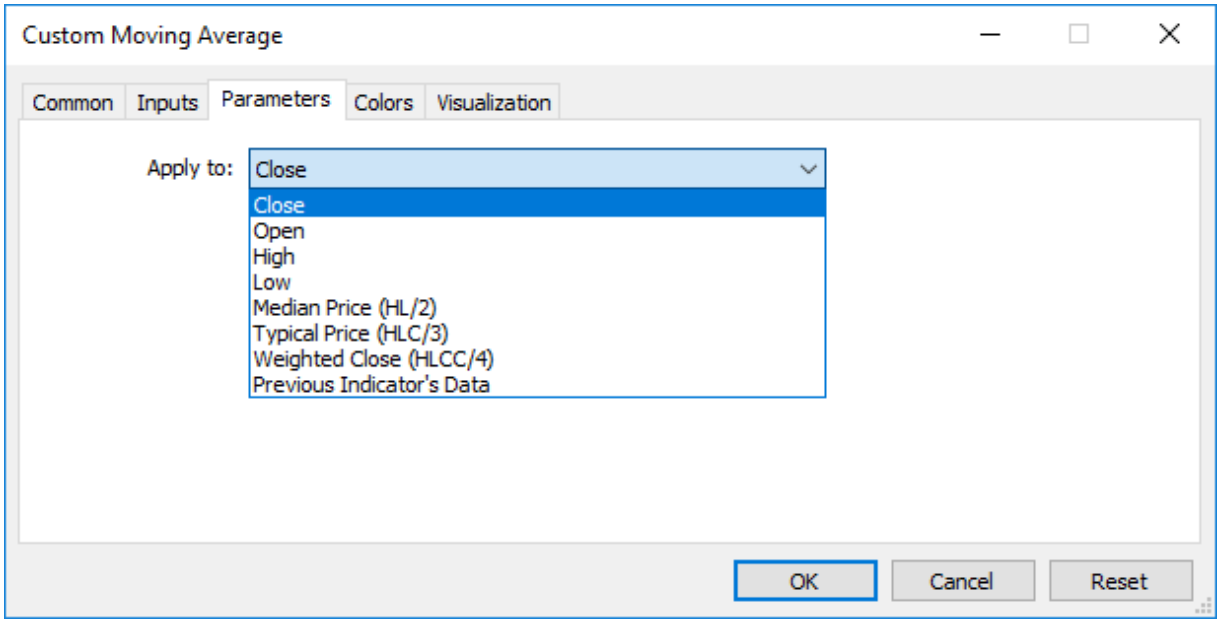
```
int OnCalculate (const int rates_total,      // taille du tableau price[]
                 const int prev_calculated,  // nombre de barres gérées lors de l'appe
                 const int begin,            // indice de début des données significat
```

© 2000-2024, MetaQuotes Ltd.

```
                    const double& price[]        // tableau à calculer
   );
```

[2]As the price[] array, one of timeseries or a calculated buffer of some indicator can be passed. Pour déterminer la direction de l'indexation du tableau price[], appelez ArrayGetAsSeries(). Afin de ne pas dépendre des valeurs par défaut, vous devez appeler la fonction ArraySetAsSeries() pour ces tableaux de façon inconditionnelle.

Les séries de temps nécessaires ou un indicateur devant être utilisé comme le tableau price[] peuvent être sélectionnés par l'utilisateur dans l'onglet "Paramètres" lors du lancement de l'indicateur. Pour cela, vous devez sélectionner l'entrée nécessaire dans la liste déroulante du champ "Appliquer à".



Pour recevoir les valeurs d'un indicateur dans d'autres programmes mql5, la fonction iCustom() est utilisée, et retourne le handle de l'indicateur pour les opérations à suivre. Vous pouvez également spécifier le tableau price[] approprié ou le handle d'un autre indicateur. Ce paramètre doit être le dernier dans la liste des variables d'entrée de l'indicateur.

Exemple :

```
void OnStart()
  {
//---
   string terminal_path=TerminalInfoString(STATUS_TERMINAL_PATH);
   int handle_customMA=iCustom(Symbol(),PERIOD_CURRENT, "Custom Moving Average",13,0,
   if(handle_customMA>0)
      Print("handle_customMA = ",handle_customMA);
   else
      Print("Impossible d'ouvrir le fichier (ou n'est pas un fichier EX5) '"+terminal_
  }
```

Dans cet exemple, le dernier paramètre passé esst la valeur PRICE_TYPICAL (de l'énumération ENUM_APPLIED_PRICE), qui indique que l'indicateur sera construit sur les prix typiques obtenus comme (High+Low+Close)/3. Si ce paramètre n'est pas spécifié, l'indicateur est construit sur les valeurs PRICE_CLOSE, c'est à dire les prix de clôture de chaque barre.

© 2000-2024, MetaQuotes Ltd.

Un autre exemple montrant le passage du handle de l'indicateur comme le dernier paramètre pour spécifier le tableau price[], est donné dans la description de la fonction iCustom().

La seconde forme est prévue pour tous les autres indicateurs dans lesquels plus d'une séries de données est utilisée pour les calculs.

```
int OnCalculate (const int rates_total,      // taille de la série de données en entré
                 const int prev_calculated,  // nombre de barres gérées lors de l'appe
                 const datetime& time[],     // Heure
                 const double& open[],       // Open
                 const double& high[],       // High
                 const double& low[],        // Low
                 const double& close[],      // Close
                 const long& tick_volume[],  // Volume du Tick
                 const long& volume[],       // Volume Réel
                 const int& spread[]         // Spread
   );
```

Les paramètres de open[], high[], low[] et close[] contiennent les tableaux des prix open, high, low et close de la période courante. Le paramètre time[] contient in tableau avec les valeurs des heures open, le paramètre spread[] est un tableau contenant l'historique des spreads (si un if spread est fourni pour l'instrument tradé). Les paramètres volume[] et tick_volume[] contiennent l'historique des volumes de trade et de tick respectivement.

Pour déterminer la direction d'indexation de time[], open[], high[], low[], close[], tick_volume[], volume[] et spread[], appelez ArrayGetAsSeries(). Afin de ne pas dépendre des valeurs par défaut, vous devez appeler la fonction ArraySetAsSeries() pour ces tableaux de façon inconditionnelle.

Le premier paramètre rates_total contient le nombre de barres disponibles pour le calcul de l'indicateur et correspond au nombre de barres disponibles dans le graphique.

Nous devons noter la connexion entre la valeur de retour de OnCalculate() et le deuxième paramètre d'entrée prev_calculated. Pendant l'appel de la fonction, le paramètre prev_calculated contient la valeur **retournée** par OnCalculate() pendant l'appel **précédent**. Ceci permet d'avoir des algorithmes plus économiques pour le calcul de l'indicateur afin d'éviter les calculs répétés pour les barres qui n'ont pas changé depuis la dernière exécution de cette fonction.

Pour cela, il est habituellement suffisant de retourner la valeur du paramètre rates_total, qui contient le nombre de barre dans l'appel actuel de la fonction. Si depuis le dernier appel à OnCalculate() les prix ont changé (un historique plus profond téléchargé ou les manques dans l'historique sont remplis)), la valeur du paramètre d'entrée prev_calculated sera mis à zéro par le terminal.

Note : si OnCalculate retourne zéro, alors les valeurs de l'indicateur ne sont pas affichées dans la fenêtre des Données dans le terminal client.

Pour mieux le comprendre, il est utile de démarrer l'indicateur dont le code est ci-dessous.

Example d'Indicateur :

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
```

© 2000-2024, MetaQuotes Ltd.

```
//---- dessin d'une ligne
#property indicator_label1  "Ligne"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDarkBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- buffers de l'indicateur
double         LineBuffer[];
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime& time[],
                const double& open[],
                const double& high[],
                const double& low[],
                const double& close[],
                const long& tick_volume[],
                const long& volume[],
                const int& spread[])
  {
//--- Récupère le nombre de barres disponibles pour le symbole courant et la période c
   int bars=Bars(Symbol(),0);
   Print("Barres = ",bars,", rates_total = ",rates_total,",  prev_calculated = ",prev_
   Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- retourne la valeur de prev_calculated pour le prochain appel
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Voir aussi

Exécuter des Programmes, Evènements du Terminal Client, Travailler avec des Evènements

© 2000-2024, MetaQuotes Ltd.

Variables

Déclarer des Variables

Les variables doivent être déclarées avec d'être utilisées. Les noms uniques sont utiliser pour identifier les variables. Pour déclarer une variable, vous devez spécifier son type et un nom unique. La déclaratio d'une variable n'est pas une opération.

Les types simples sont :

- char, short, int, long, uchar, ushort, uint, ulong - entiers ;
- color - entier représentant une couleur RGB ;
- datetime - date et heure, un entier non signé contenant le nombre de secondes depuis le 1er janvier 1970, 00h00 ;
- bool - les valeurs booléennes *true* et *false* ;
- double - nombres à virgule flottante à double précision ;
- float - nombres à virgule flottante à simple précision ;
- string - chaînes de caractères.

Exemples :

```
string szInfoBox;
int    nOrders;
double dSymbolPrice;
bool   bLog;
datetime tBegin_Data   = D'2004.01.01 00:00';
color    cModify_Color = C'0x44,0xB9,0xE6';
```

Types complexes ou composés :

Les structures sont des types de données composés, construits en utilisant d'autres types.

```
struct MyTime
  {
   int hour;    // 0-23
   int minute;  // 0-59
   int second;  // 0-59
  };
...
MyTime strTime; // Variable du type de la structure MyTime déclarée précédemment
```

Vous ne pouvez pas déclarer de variables du type de la structure tant que vous n'avez pas déclaré la structure.

Tableaux

Un tableau est une suite indexée de données de même type :

```
int    a[50];        // Tableau à 1 dimension de 50 entiers.
double m[7][50];     // Tableau à 2 dimensions de 7 tableaux,
                     // chacun d'eux ayant 50 nombres.
```

© 2000-2024, MetaQuotes Ltd.

```
MyTime t[100];       // Tableau content 100 éléments de type MyTime
```

L'indice d'un tableau ne peut être qu'un entier. Il n'est pas possible d'avoir plus de 4 dimensions dans un tableau. L'indexation des éléments d'un tableau commence à 0. Le dernier élément d'un tableau à 1 dimension est à l'indice (taille du tableau)-1. Cela signifie que l'appel au dernier élément d'un tableau t fait de 50 entiers sera t[49]. C'est identique pour les tableaux à plusieurs dimensions : une dimension est indexée de 0 à la taille de la dimension -1. Le dernier élément d'un tableau à 2 dimension de l'exemple apparaîtra sous la forme m[6][49].

Les tableaux statiques ne peuvent pas être représentés comme des séries temporelles (timeseries), c'est à dire que la fonction ArraySetAsSeries(), qui définit l'accès aux éléments du tableau de la fin vers le début, ne peut pas être appliquée. Si vous souhaitez donner accès à un tableau de la même façon que dans les timeseries, utilisez l'objet tableau dynamique.

Si une tentative d'accéder en dehors du tableau est faite, le système génèrera une erreur critique et le programme sera stoppé.

Méthodes intégrées pour travailler avec des tableaux

Les fonctions de la section Fonctions sur les Tableaux peuvent être utilisées pour manipuler les tableaux, tout comme les fonctions intégrées :

Méthode	Analogue à	Description
void array.Fill(const scalar value, const int start_pos=0, const int count=-1);	ArrayFill, ArrayInitialize	Remplit le tableau avec la valeur spécifiée
void array.Free();	ArrayFree	Libère le buffer du tableau et définit la taille de la dimension zéro sur 0 (zéro)
int array.Resize(const int range0_size, const int reserve); int array.Resize(const int range_sizes[], const int reserve);	ArrayResize	Définit la nouvelle taille de la première dimension du tableau
int array.Print();	ArrayPrint	Ecrit les valeurs du tableau de type simple dans le journal
int array.Size(const int range=-1);	ArraySize, ArrayRange	Renvoie le nombre d'éléments dans le tableau entier (range=-1) ou dans la dimension de tableau spécifiée
bool array.IsDynamic();	ArrayIsDynamic	Vérifie si le tableau est dynamique
bool array.IsIndicatorBuffer();		Vérifie si le tableau est un buffer d'indicateur
bool array.IsSeries();	ArrayIsSeries	Vérifie si le tableau est une timeserie (série temporelle)
bool array.AsSeries();	ArrayGetAsSeries	Vérifie la direction d'indexation du tableau

© 2000-2024, MetaQuotes Ltd.

Méthode	Analogue à	Description
bool array.AsSeries(const bool as_series);	ArraySetAsSeries	Définit le sens d'indexation dans le tableau
int array.Copy(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCopy	Copie les valeurs du tableau dans un autre tableau
int array.Compare(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCompare	Renvoie le résultat de la comparaison de deux tableaux de types simples ou de structures personnalisées
int array.Insert(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayInsert	Insère le nombre spécifié d'éléments du tableau source dans le tableau destination à partir de l'index spécifié
int array.Remove(const int start_pos, const int count);	ArrayRemove	Supprime le nombre spécifié d'éléments du tableau en commençant à l'index spécifié
int array.Reverse(const int start_pos, const int count);	ArrayReverse	Inverse le nombre spécifié d'éléments dans le tableau en commençant à l'index spécifié
bool array.Swap(array& arr[]);	ArraySwap	Echange le contenu du tableau avec le contenu d'un autre tableau dynamique de même type
void array.Sort(sort_function);	ArraySort	Trie les tableaux numériques par la première dimension
int array.Search(scalar value, search_function);	ArrayBsearch	Renvoie l'indice du premier élément détecté dans la première dimension du tableau
int array.Find((scalar value, search_function);		Effectue une recherche dans le tableau à l'aide de la fonction passée et renvoie l'index du premier élément détecté
array array.Select(scalar value, search_function);		Performs a search in the array using the passed function and returns the array with all detected elements

Spécificateurs d'Accès

Les spécificateurs d'accès définissent comment le compilateur peut accéder aux variables, aux membres des structures ou des classes.

© 2000-2024, MetaQuotes Ltd.

Le spécificateur const déclare une variable comme constante et ne permet pas de changer cette pendant l'exécution. Une initialisation simple d'une variable est autorisée lors de sa déclaration.

Exemple :

```
int OnCalculate (const int rates_total,      // taille du tableau price[]
                 const int prev_calculated,  // nombre de barres gérées lors de l'appe
                 const int begin,            // indice de début des données significat
                 const double& price[]       // tableau à calculer
   );
```

Pour accéder aux membres des structures et des classes, utilisez les qualificateurs suivants :

- public - permet un accès non restreint à la variable ou à la méthode de la classe
- protected - permet l'accès depuis les méthodes de cette classe ainsi que depuis les méthodes des classes héritées publiquement. Tout autre accès est impossible ;
- private - permet l'accès aux variables et aux méthodes de la classe seulement depuis les méthodes de la même classe.
- virtual - ne s'applique qu'aux méthodes de classe (mais pas aux méthodes de structure) et indique au compilateur que cette méthode doit être placée dans la table des fonctions virtuelles de la classe.

Classes de Stockage

Il existe 3 types de classes de sotckage : static, input et extern. Ces modificateurs d'une classe de stockage indiquent explicitement au compilateur que les variables correspondantes sont distribuées dans une zone mémoire pré-allouée, qui est appelée la zone globale. En outre, ces modificateurs indiquent un traitement spécial aux données de la variable. Si une variable déclarée à un niveau global n'est pas statique, la mémoire de cette variable est allouée automatiquement sur la pile du programme. La libération de la mémoire allouée à un tableau non statique est également effectuée automatiquement lors de la sortie de la portée du bloc dans lequel le tableau est déclaré.

Voir aussi

Types de Données, Encapsulation et Extension des Types,Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets, Membres Statiques d'une Class

© 2000-2024, MetaQuotes Ltd.

Variables Locales

une variable déclarée dans une fonction est local. La portée d'une variable locale est limitée à la portée de la fonction dans laquelle elle est déclarée. Une variable locale peut être initialisée par le résultat de n'importe quelle expression. Chaque appel à la fonction initialise une variable locale. Les variables locales sont stockées dans la zone de mémoire correspondant à la fonction.

Exemple :

```
int somefunc()
  {
   int ret_code=0;
   ...
   return(ret_code);
  }
```

La portée d'une variable est la partie du progamme dans laquelle cette variable peut être utilisée. Les variables déclarées dans un bloc (au niveau interne) ont le bloc comme portée. La portée du block commence avec la déclaration de la variable et se termine avec la dernière parenthèse fermante du bloc.

Les variables locales déclarées au début d'une fonction ont également la portée du bloc, tout comme les paramètres de fonction qui sont des variables locales. Tout bloc peut contenir des déclarations de variables. Si les blocs sont imbriqués et que l'identifiant du bloc externe a le même nom que l'identifiant dans le bloc interne, l'identifiant du bloc externe est caché, jusqu'à la fin du bloc interne.

Exemple :

```
void OnStart()
  {
//---
   int i=5;       // variable locale de la fonction
     {
      int i=10;  // variable de la fonction
      Print("Dans le bloc i = ",i); // le résultat est i=10;
     }
   Print("En dehors du bloc i = ",i);  // le résultat est i=5;
  }
```

Cela signifie que tant que le bloc interne est en cours d'exécution, il voit les valeurs de ses propres identifiants locaux, pas les valeurs des identifiants ayant des noms identiques dans le bloc externe.

Exemple :

```
void OnStart()
  {
//---
   int i=5;       // variable locale de la fonction
   for(int i=0;i<3;i++)
      Print("Dans la boucle for i = ",i);
   Print("En dehors du bloc i = ",i);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
/*  Résultat de l'exécution
Dans la boucle for i = 0
Dans la boucle for i = 1
Dans la boucle for i = 2
En dehors du bloc i = 5
*/
```

Les variables locales déclarées comme static ont la portée du bloc, bien qu'elles existent depuis le lancement du programme.

Pile (stack)

Dans chaque programme MQL5, une zone spéciale de la mémoire, appelée pile, est allouée pour stocker les variables des fonctions locales qui sont créées automatiquement. Une pile est allouée pour toutes les fonctions, sa taille est par défaut pour les indicateurs est égale à 1 Mo. Dans les Expert Advisors et les scripts, la taille de la pile peut être gérée en utilisant la directive de compilation #property stacksize (qui définit la taille de la pile en octets), une mémoire de 8 Mo est allouée par défaut pour la pile.

Les variables locales statiques sont stockées au même endroit que les autres variables statiques et globales - une zone spéciale de la mémoire qui existe séparément de la pile. Les variables créées dynamiquement utilisent égaement une zone mémoire séparée de la pile.

Pour chaque appel d'une fonction, une place est allouée sur la pile pour les variables internes non statiques. A la sortie de la fonction, la mémoire est à nouveau disponible.

Si une deuxième fonction est appelée depuis la première, alors la seconde fonction occupe la taille nécessaire dans la mémoire restante de la pile pour ses propres variables. Lors de l'utilisation de fonctions inclues, la mémoire de la pile est donc occupée de façon séquentielle pour chaque fonction. Ceci peut provoquer un manque de mémoire pendant l'un des appels de fonction, cette situation est appelée un débordement de pile (stack overflow).

Pour les grosses données locales, il est donc mieux d'utiliser de la mémoire dynamique - lors de l'entrée dans une fonction, allouez la mémoire nécesaire dans le système pour les besoins locaux (new, ArrayResize()), et à la fin de la fonction, libérez la mémoire (delete, ArrayFree()).

Voir aussi

Types de Données, Encapsulation et Extensision des Types,Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Crée et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Paramètres Formels

Les paramètres passés à une fonction sont locaux. La portée est le bloc de la fonction. Les paramètres formels doivent avoir des noms différents des variables externes et des variables locales définies dans une fonction. Certaines valeurs peuvent être assignées aux paramètres formels dans le bloc de la fonction. Si un paramètre formel est déclaré avec le modificateur const, sa valeur ne peut pas être changée dans la fonction.

Exemple :

```
void func(const int & x[], double y, bool z)
  {
   if(y>0.0 && !z)
      Print(x[0]);
   ...
  }
```

Les paramètres formels peuvent être initialisés avec des constantes. Dans ce cas, la valeur d'initialisation est considérée comme la valeur par défaut. Les paramètres suivant les paramètres initialisés, doivent également être initialisés.

Exemple :

```
void func(int x, double y = 0.0, bool z = true)
  {
   ...
  }
```

Lors de l'appel à une telle fonction, les paramètres initialisés peuvent être omis, les valeurs par défaut se substituant à eux.

Exemple :

```
func(123, 0.5);
```

Les paramètres de types simples sont passés par valeur, c'est à dire que les modifications de la variable locale correspondante de ce type dans la fonction appelée ne seront pas reflétées dans la fonction appelante. Les tableaux de n'importe quel type et les données de type structure sont toujours passés par référence. S'il est nécessaire d'interdire la modification du tableau ou du contenu d'une structure, les paramètres de ces types doivent être déclarés avec le mot-clé *const*.

Il est possible de passer les paramètres de types simples par référence. Dans ce cas, la modification de ces paramètres dans la fonction appelante affectera les variables correspondantes passées par référence. Pour indiquer qu'un paramère est passé par référence, indiquez le modificateur & après le type de la donnée.

Exemple :

```
void func(int& x, double& y, double & z[])
  {
   double calculated_tp;
   ...
   for(int i=0; i<OrdersTotal(); i++)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       if(i==ArraySize(z))        break;
       if(OrderSelect(i)==false) break;
       z[i]=OrderOpenPrice();
      }
    x=i;
    y=calculated_tp;
   }
```

Les paramètres passés par référence ne peuvent pas être initialisés par des valeurs par défaut.

Un maximum de 64 paramètres peuvent être passés à une fonction.

Voir aussi

Variables d'Entrée, Types de Données, Encapsulation et Extension des Types, Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Variables Statiques

La classe de stockage static définit une variable statique. Le modificateur static est indiqué avant le type de données.

Exemple :

```
int somefunc()
  {
   static int flag=10;
   ...
   return(flag);
  }
```

Une variable statique peut être initialisée par une constante ou par une expression constante correspondant à son type, contrairement à une simple variable locale, qui peut être initialisée par n'importe quelle expression.

Les variables statiques existent à partir de l'exécution du programme et ne sont initialisées qu'une seule fois avant l'appel de la fonction spécialiséeOnInit(). Si les valeurs initiales ne sont pas spécifiées, les variables statiques sont initialisées à zéro.

Les variables locales déclarées avec le mot-clé *static* conservent leurs valeurs tout au long du cycle de vie de la fonction. A chaque nouvel appel de la fonction, ces variables locales contiennent les valeurs qu'elles avaient à l'appel précédent.

Toute variable dans un bloc, excepté les paramètres formels d'une fonction, peut être définie comme statique. Si une variable déclarée au niveau local n'ests pas statique, la mémoire est allouée automatiquement sur la pile du programme.

Exemple :

```
int Counter()
  {
   static int count;
   count++;
   if(count%100==0) Print("La fonction Counter a été appelée ",count," fois");
   return count;
  }
void OnStart()
  {
//---
   int c=345;
   for(int i=0;i<1000;i++)
     {
      int c=Counter();
     }
   Print("c =",c);
  }
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Types de Données, Encapsulation et Extension des Types, Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets, Membres Statiques de Classe

© 2000-2024, MetaQuotes Ltd.

Variables Globales

Les variables globales sont créée en plaçant leurs déclarations en dehors des descriptions de fonction. Les variables globales sont définies au même niveau que les fonctions, c'est à dire qu'elles ne sont locales à aucun bloc.

Exemple :

```
int GlobalFlag=10;   // Variable globale
int OnStart()
  {
   ...
  }
```

La portée des variables globales est le programme entier. Les variables globales sont accessibles de toutes les fonctions définies dans le programme. Elles sont initialisées à zéro sauf si une autre valeur initiale a été explicitement définie. Une variable globale ne peut être initialisée que par une constante ou par une expression constante correspondant à son type.

Les variables globales ne sont initialisées qu'une seule fois après le chargement du programme dans le terminal client et avant le premier traitement de l'évènement Init event. Pour les variables globales représentant des objets de classe, les constructeurs correspondant sont appelés pendant leur initialisation. Dans les scripts, les variables globales sont initialisées avant le traitement de l'évènement Start.

Note : Les variables déclarées au niveau globales ne doivent pas être confondues avec les variables globales du terminal client qui sont accessibles avec les fonctions GlobalVariable...().

Voir aussi

Types de Données, Encapsulation et Extensision des Types,Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Crée et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Variables d'Entrées

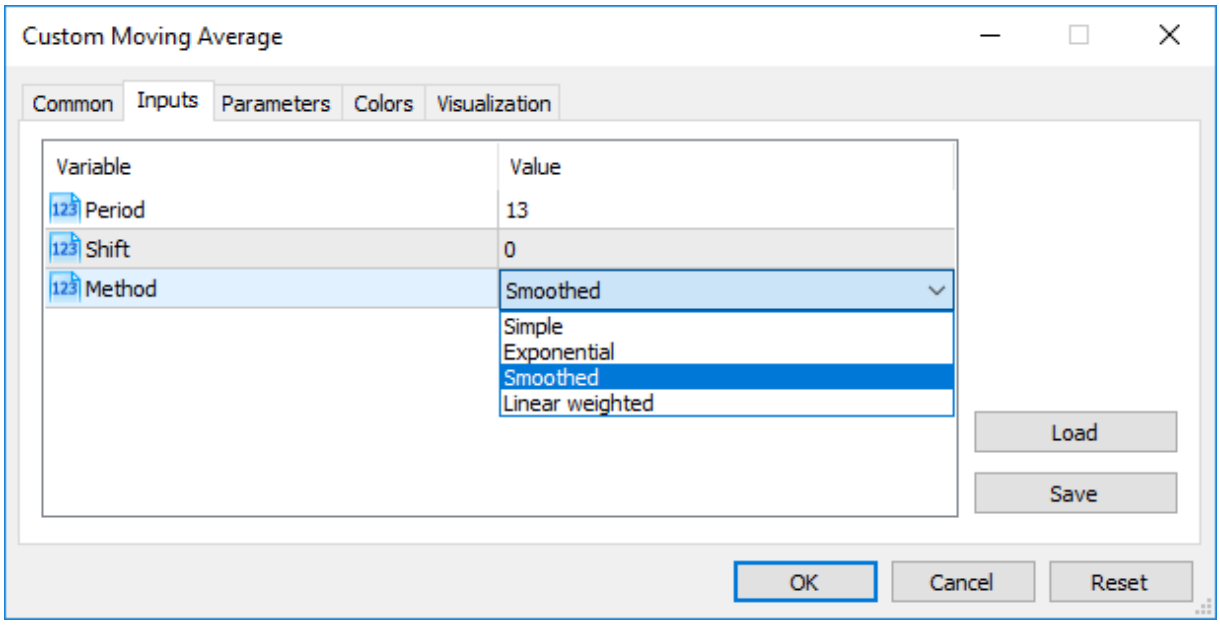
La classe de stockage input définit une variable externe (variable d'entrée). Le modificateur *input* est indiqué avant le type de données. Une variable avec le modificateur input ne peut pas être changée dans les programmes mql5, ce type de variables n'est accessible qu'en lecture seule. Les valeurs des variables d'entrée ne peut être changée que par l'utilisateur et depuis la fenêtre des propriétés du programme. Les variables externes sont toujours réinitialisées immédiatement avant l'appel à OnInit().

La longueur maximale des noms de variables d'entrée est de 63 caractères. Pour le paramètre d'entrée de type chaîne, la longueur de valeur **maximum** (longueur de chaîne) peut être de 191 à 253 caractères (voir la Note). La longueur minimale est de 0 caractère (la valeur n'est pas définie).

Exemple :

```
//--- input parameters
input int            MA_Period=13;
input int            MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMMA;
```

Les variables d'entrée déterminent les paramètres d'entrée d'un programme. Elles sont disponibles dans la fenêtre Propriétés d'un programme.

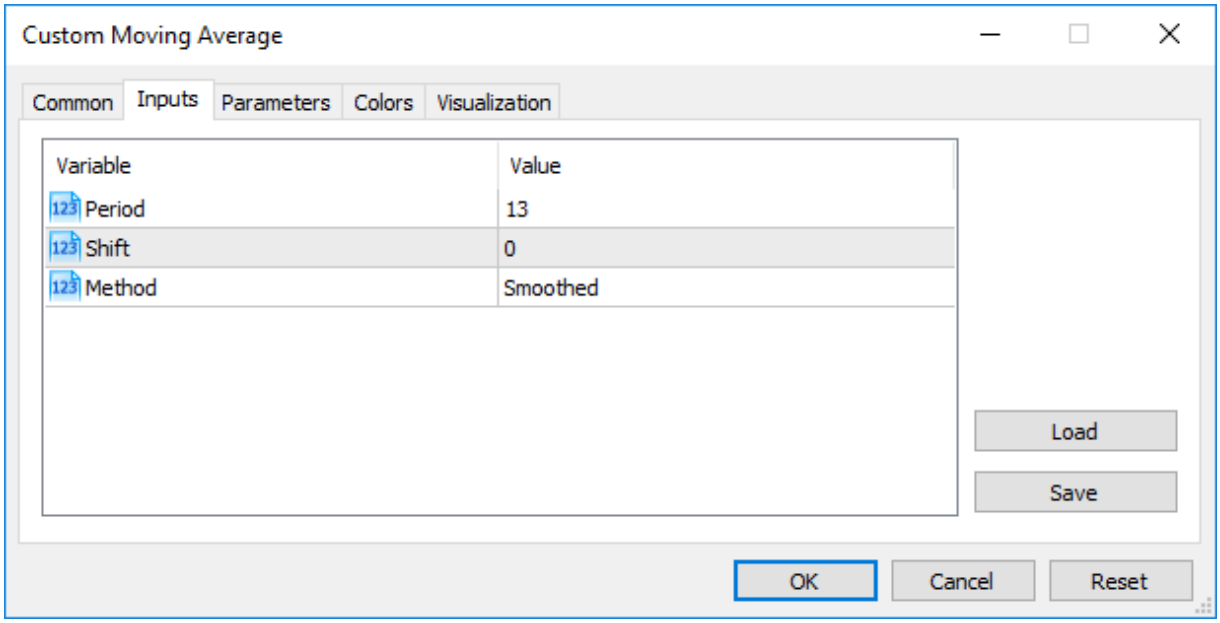


Il existe une autre façon de définir à quoi ressembleront les paramètres d'entrée dans l'onglet Inputs. Pour cela, placez un commentaire après la description d'un paramètre d'entrée sur la même ligne. De cette façon, vous pouvez donner des noms plus compréhensibles pour les utilisateurs aux variables d'entrée.

Exemple :

```
//--- input parameters
input int            InpMAPeriod=13;         // Période de lissage
input int            InpMAShift=0;           // Décalage de la ligne horizontale
input ENUM_MA_METHOD InpMAMethod=MODE_SMMA;  // Méthode de lissage
```

© 2000-2024, MetaQuotes Ltd.



Note : Les tableaux et variables de types complexes ne peut pas servir de variables d'entrée.

Note : La longueur de la chaîne du commentaire des variables d'entrée ne peut pas être supérieure à 63 caractères.

Note : Pour les variables d'entrée de type chaîne, la limitation de la longueur de la valeur (longueur de la chaîne) est fixée par les conditions suivantes :

- la valeur du paramètre est représentée par la chaîne "nom_paramètre = valeur_paramètre" ('=' est pris en compte),
- longueur de représentation maximale de 255 caractères (*total_length_max*= 255 ou 254 caractères hors '='),
- longueur maximale du paramètre de chaîne *parameter_name_length* = 63 caractères.

Ainsi, la taille de chaîne maximale pour un paramètre de chaîne est calculée à l'aide de l'équation :

```
parameter_value_length=total_length_max-parameter_name_length = 254 - parameter_name_
```

Cela fournit la taille de chaîne maximale de 191 (paramètre_nom_longueur = 63) à 253 caractères (paramètre_nom_longueur = 1).

Passage des Paramètres Lors de l'Appel à des Indicateurs depuis les Programmes MQL5

Les Indicateurs Personnalisés sont appelés avec la fonction iCustom(). Après le nom de l'indicateur, les paramètres doivent être en accord strict avec la déclaration des variables d'entrée de cet indicateur. Si le nombre de paramètres indiqués est inférieur au nombre de variables déclarées dans l'indicateur appelé, les paramètres manquants sont remplis avec les valeurs spécifiées pendant la déclaration des variables.

Si l'indicateur utilise la fonction OnCalculate du premier type (c'est à dire que l'indicateur est calculé en utilisant le même tableau de données), alors l'une des valeurs de ENUM_APPLIED_PRICE ou le handle

© 2000-2024, MetaQuotes Ltd.

d'un autre indicateur doit être utilisé comme dernier paramètre lors de l'appel à cet indicateur. Tous les paramètres correspondants aux variables d'entrée doivent être indiqués clairement.

Enumérations comme Paramètres d'Entrée

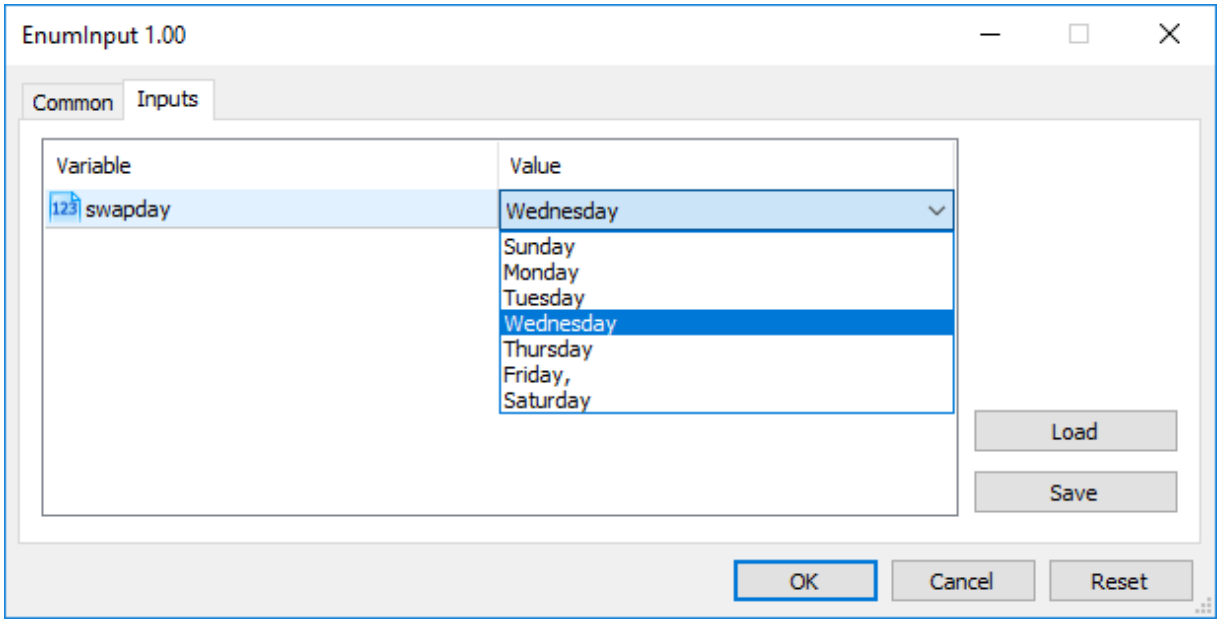
Les variables définies par l'utilisateur ainsi que les énumérations intégrées en MQL5 peuvent être utilisées comme variables d'entrée (paramètres d'entrée pour les programmes mql5). Par exemple, nous pouvons créer l'énumération dayOfWeek, décrivant les jours de la semaine et utiliser la variable d'entrée pour spécifier un jour particulier de la semaine, non pas comme un chiffre, mais d'une façon plus courante.

Exemple :

```
#property script_show_inputs
//--- jour de la semaine
enum dayOfWeek
  {
   S=0,      // Dimanche
   M=1,      // Lundi
   T=2,      // Mardi
   W=3,      // Mercredi
   Th=4,     // Jeudi
   Fr=5,     // Vendredi,
   St=6,     // Samedi
  };
//--- input parameters
input dayOfWeek swapday=W;
```

Pour permettre à un utilisateur de sélectionner la valeur désirée dans la fenêtre des propriétés pendant le démarrage du script, nous utilisons la commande du préprocesseur #property script_show_inputs. Nous lançons le script et pouvons choisir l'une des valeurs de l'énumération dayOfWeek dans la liste. Nous lançons le script EnumInInput et allons dans l'onglet Inputs. Par défaut, la valeur de swapday (jour de facturation triple du swap) est mercredi (W = 3), mais il est possible de spécifier n'importe quelle valeur pour changer le fonctionnement du programme.

© 2000-2024, MetaQuotes Ltd.



Le nombre de valeurs possibles d'une énumération est limité. Pour sélectionner une valeur d'entrée, une liste déroulante est utilisée. Les noms mnémoniques pour les membres de l'énumération sont utilisés pour les valeurs affichées dans la liste. Si un commentaire est associé avec un nom mnémonique, comme affiché dans l'exemple, le contenu du commentaire est utilisé au lieu du nom mnémonique.

Chaque valeur de l'énumération dayOfWeek a une valeur de 0 à 6, mais dans la liste des paramètres, les commentaires spécifiés pour chaque valeur seront affichés. Ceci fournit une flexibilité supplémentaire pour écrire des programmes avec des descriptions claires des paramètres d'entrée.

Variables avec le Modificateur Input

Les variables avec le modificateur input permettent non seulement de définir des paramètres externes lors du lancement des programmes, mais sont également nécessaires pour optimiser les stratégies de trading dans le Strategy Tester. Chaque variable d'entrée peut être utilisée dans l'optimisation, sauf celles de type chaîne de caractères.

Il est parfois nécessaire d'exclure certains paramètres externes du programme de la zone de toutes les passes du tester. Le modificateur de mémoire sinput a été introduit pour ce type de cas. sinput signifie déclaration de variable externe statique (sinput = static input). Cela signifie que la déclaration suivant dans le code d'un Expert Advisor

```
sinput        int layers=6;    // Nombre de couches
```

sera équivalent à la déclaration complète

```
static input int layers=6;    // Nombre de couches
```

La variable déclarée avec le modificateur sinput est un paramètre d'entrée du programme MQL5. La valeur de ce paramètre peut être changée au lancement du programme. Cependant, cette variable n'est pas utilisée dans l'optimisation des paramètres d'entrée. En d'autres termes, ses valeurs ne sont

© 2000-2024, MetaQuotes Ltd.

pas énumérées lors de la recherche du meilleur ensemble de paramètres remplissant une condition spécifiée.

Strategy Tester

Variable	Value	Start	Step	Stop	Steps
Number of layers	6				
Neurons in a layer	**30**	**30**	**1**	**300**	**271**
Number of bars to be analyzed	**13**	**13**	**1**	**130**	**118**
Forecast horizon	**2**	**2**	**1**	**20**	**19**
Network type	0	0	1	10	
					607582

Settings | Inputs | Optimization Results | Agents | Journal

L'Expert Advisor montré ci-dessus a 5 paramètres externes. "Nombre de couches" est déclaré comme étant sinput et est égal à 6. Ce paramètre ne peut pas être changé pendant une optimisation de la stratégie de trading. Nous pouvons spécifier une valeur désirée à utiliser par la suite. Les champs Démarrer, Pas et Stop ne sont pas disponible pour ce type de variable.

Les utilisateurs ne seront donc pas capables d'optimiser ce paramètre une fois que nous aurons spécifié le modificateur sinput pour la variable. En d'autres termes, les utilisateurs du terminal ne pourront pas lui définir des valeurs de départ et de fin dans le Strategy Tester pour une énumération automatique dans l'intervalle spécifié pendant l'optimisation.

Cependant, il existe une exception à cette règle : les variables sinput peuvent varier dans les tâches d'optimisation en utilisant la fonction ParameterSetRange(). Cette fonction a été introduite spécifiquement pour le contrôle par le programme des valeurs disponibles pour n'importe quelle variable input incluant celles déclarées comme static input (sinput). La fonction ParameterGetRange() permet de recevoir les valeurs des variables d'entrée lorsque l'optimisation est lancée (dans le gestionnaire OnTesterInit()) et de réinitialiser une valeur de changement de pas et un intervalle dans lequel seront les valeurs du paramètre optimisé.

De cette façon, combiner le modificateur sinput et deux fonctions utilisant les paramètres d'entrée, permet de créer des règles flexibles pour définir les intervalles d'optimisation des paramètres d'entrée qui dépendent des valeurs d'autres paramètres d'entrée.

Arranger les paramètres d'entrée

Pour travailler plus efficacement avec les programmes MQL5, les paramètres d'entrée peuvent être regroupés en blocs grâce au mot-clé group. Celui-ci permet une séparation visuel de certains paramètres suivant leur logique.

```
input group          "Nom du groupe"
input int            variable1 = ...
input double         variable2 = ...
input double         variable3= ...
```

Après une déclaration comme celle-ci, tous les paramètres d'entrée sont liés visuellement dans un même groupe, simplifiant la configuration des paramètres pour les utilisateurs MQL5 lorsqu'ils sont lancés sur un graphique ou dans le Strategy Tester. La spécification de chaque groupe est valide jusqu'à la prochaine déclaration de groupe :

© 2000-2024, MetaQuotes Ltd.

```
input group           "Nom du groupe #1"
input int             group1_var1 = ...
input double          group1_var2 = ...
input double          group1_var3 = ...

input group           "Nom du groupe #2
input int             group2_var1 = ...
input double          group2_var2 = ...
input double          group2_var3 = ...
```

Un exemple d'EA présentant les blocs d'entrées séparés suivant leur objectif :

```
input group           "Signal"
input int             ExtBBPeriod    = 20;        // Période des Bandes de Bollinger
input double          ExtBBDeviation= 2.0;        // déviation
input ENUM_TIMEFRAMES ExtSignalTF=PERIOD_M15;     // Période BB

input group           "Tendance"
input int             ExtMAPeriod    = 13;        // Période de la Moyenne Mobile
input ENUM_TIMEFRAMES ExtTrendTF=PERIOD_M15;      // Période MM

input group           "ReglesDeSortie"
input bool            ExtUseSL       = true;      // utilise le StopLoss
input int             Ext_SL_Points = 50;         // StopLoss en points
input bool            ExtUseTP       = false;     // utilise le TakeProfit
input int             Ext_TP_Points = 100;        // TakeProfit en points
input bool            ExtUseTS       = true;      // utilise le Trailing Stop
input int             Ext_TS_Points = 30;         // Trailing Stop en points

input group           "MoneyManagement"
sinput double         ExtInitialLot = 0.1;        // valeur initiale du lot
input bool            ExtUseAutoLot = true;       // calcul automatique du lot

input group           "Auxiliaire"
sinput int            ExtMagicNumber = 123456;    // Magic Number de l'EA
sinput bool           ExtDebugMessage= true;      // affiche les messages de debug
```

Lors du lancement d'un tel EA dans le Strategy Tester, vous pouvez double-cliquer sur le nom du groupe pour réduire/développer le bloc d'entrées, puis cocher la case du groupe pour sélectionner tous ses paramètres à optimiser.

© 2000-2024, MetaQuotes Ltd.

Strategy Tester

Select expert... | View previous optimization results

Variable	Value	Start	Step	Stop	Steps
Signal					
☑ Bollinger Bands period	20	20	10	60	5
☑ deviation	2	2	0.2	3	6
☐ BB timeframe	15 Minutes	current		30 Minutes	
Trend					
☑ Moving Average period	13	13	1	20	8
☐ MA timeframe	15 Minutes	current		1 Hour	
ExitRules					
☑ use StopLoss	true	false		true	2
☑ StopLoss in points	50	50	10	100	6
☑ use TakeProfit	false	false		true	2
☑ TakeProfit in points	100	100	50	300	5
☑ use Trailing Stop	true	false		true	2
☑ Trailing Stop in points	30	30	10	70	5
MoneyManagement					
☐ initial lot value	0.1				
☑ automatic lot calculation	true	false		true	2
Auxiliary					
☐ EA Magic Number	123456				
☐ print debug messages	true				
					576000

Overview | Settings | Inputs | Backtest | Graph | Optimization Results | Agents | Journal | Start

Voir aussi

iCustom, Enumérations, Propriétés des Programmes

© 2000-2024, MetaQuotes Ltd.

Variables Externes

Le mot-clé extern est utilisé pour déclarer les identifiants des variables comme identifiants de la classe de stockage statique avec un cycle de vie global. Ces variables existent depuis le lancement du programme et leur mémoire est allouée et initialisée immédiatement après le démarrage du programme.

Vous pouvez créer des programmes faits de plusieurs fichiers source ; dans ce cas une directive #include du pré-processeur est utilisée. Des variables déclarées comme externes avec le même type et le même identifiant peuvent exister dans plusieurs fichiers source d'un même projet.

Lors de la compilation du projet entier, toutes les variable externes ayant le même type et le même identifiant sont associées dans une partie de la mémoire de la zone globale. Les variables externes sont utiles pour la compilation séparées des fichiers source. Les variables externes peuvent être initialisées, mais une seule fois - l'existence de plusieurs variables externes initialisées du même type et avec le même identifiant est interdite.

Voir aussi

Types de Données, Encapsulation et Extensision des Types,Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables, Crée et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Initialisation des Variables

Toute variable peut être initialisée pendant sa définition. Si une variable n'est pas initialisée explicitement, elle peut avoir n'importe quelle valeur au démarrage. L'initialisation implicite n'est pas utilisée.

Les variables globales et statiques ne peuvent être initialisées que par une constante du type correspondant ou par une expression constante. Les variables locales peuvent être initialisées par n'importe quelle expression, et pas seulement par une constante.

L'initialisation des variables globales et des variables statiques n'est effectuée qu'une seule fois. L'initialisation des variables locales est effectuée à chaque appel des fonctions correspondantes.

Exemples :

```
int    n       = 1;
string s       = "hello";
double f[]     = { 0.0, 0.236, 0.382, 0.5, 0.618, 1.0 };
int    a[4][4] = { {1, 1, 1, 1}, {2, 2, 2, 2}, {3, 3, 3, 3}, {4, 4, 4, 4 } };
//--- de tetris
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                 WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
//--- initialisation de tous les champs de la structure avec la valeur zéro
MqlTradeRequest request={};
```

La liste des valeurs des éléments du tableau doit être placée entre parenthèses. Les suites d'initialisation manquantes sont considérées comme égales à 0.

Si la taille du tableau initialisé n'est pas spécifiée, elle est déterminée par le ompilateur, suuvant la taille de la suite d'initialisation.

Exemples :

```
struct str3
  {
   int                low_part;
   int                high_part;
  };
struct str10
  {
   str3               s3;
   double             d1[10];
   int                i3;
  };
void OnStart()
  {
   str10 s10_1={{1,0},{1.0,2.1,3.2,4.4,5.3,6.1,7.8,8.7,9.2,10.0},100};
   str10 s10_2={{1,0},{},100};
   str10 s10_3={{1,0},{1.0}};
//---
   Print("1.  s10_1.d1[5] = ",s10_1.d1[5]);
```

© 2000-2024, MetaQuotes Ltd.

```
  Print("2.  s10_2.d1[5] = ",s10_2.d1[5]);
  Print("3.  s10_3.d1[5] = ",s10_3.d1[5]);
  Print("4.  s10_3.d1[0] = ",s10_3.d1[0]);
}
```

Pour les variables de type structure, l'initialisation partielle est autorisée, de même que pour les tableaux statiques (avec une taille définie implicitement). Vous pouvez initialiser un, ou plus, élément d'une structure ou d'un tableau, les éléments restants seront initialisés avec zéro dans ce cas.

Voir aussi

Types de Données, Encapsulation et Extension des Types, Portée de Visibilité et Cycle de Vie des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Portée de Visibilité et Cycle de Vie des Variables

Il existe deux types de portée : portée locale et portée globale.

Une variable déclarée en dehors de toutes les fonctions est située dans la portée globale. L'accès à ces variables peut être fait de n'importe où dans le programme. Ces variables sont situées dans la zone globale de la mémoire, leur cycle de vie coincide donc avec le cycle de vie du programme.

Une variable déclarée dans un bloc (partie du code entourée d'accolades) appartient à la portée locale. Une telle variable n'est pas visible (et donc non disponible) en dehors du bloc dans lequel elle est déclarée. La cas le plus courant de déclaration locale est une variable déclarée dans une fonction. Une variable déclarée localement est située sur la pile, et son cycle de vie est égal au cycle de vie de la fonction.

Puisque la portée d'une variable locale est le bloc dans lequel elle est déclarée, il est possible de déclarer des variables avec le même nom que des variables déclarées dans des blocs supérieurs ; de même que pour celles déclarées aux niveaux supérieurs, jusqu'au niveau global.

Exemple :

```
void CalculateLWMA(int rates_total,int prev_calculated,int begin,const double &price[]
  {
   int        i,limit;
   static int weightsum=0;
   double     sum=0;
//---
   if(prev_calculated==0)
     {
      limit=MA_Period+begin;
      //--- affecte une valeur vide aux premières barres
      for(i=0; i<limit; i++) LineBuffer[i]=0.0;
      //--- calcule la première valeur visible
      double firstValue=0;
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
      firstValue/=(double)weightsum;
      LineBuffer[limit-1]=firstValue;
     }
   else
     {
      limit=prev_calculated-1;
     }

   for(i=limit;i<rates_total;i++)
     {
      sum=0;
```

© 2000-2024, MetaQuotes Ltd.

```
      for(int j=0; j<MA_Period; j++) sum+=(MA_Period-j)*price[i-j];
      LineBuffer[i]=sum/weightsum;
     }
//---
  }
```

Faites attention à la variable i, déclarée en ligne

```
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
```

Sa portée n'est que pour la boucle for ; en dehors de cette bouel, il y a une autre variable avec le même nom déclarée au début de la fonction. De plus, la variable k est déclarée dans le corps de la boucle, sa portée est le corps de la boucle.

Les variables locales peuvent être déclarées avec le spécificateur d'accès static. Dans ce cas, le compilateur a une variable dans la zone globale de mémoire. Le cycle de vie d'une variable statique est donc égal au cycle de vie du programme. Ici, la portée d'une telle variable est limitée au bloc dans lequel elle est déclarée.

Voir aussi

Types de Données, Encapsulation et Extension des Types, Initialisation des Variables, Créer et Supprimer des Objets

© 2000-2024, MetaQuotes Ltd.

Créer et Supprimer des Objets

Une fois qu'un programme MQL5 est chargé pour être exécuté, la mémoire est allouée pour chaque variable selon son type. Selon le niveau d'accès, toutes les variables sont divisées en deux types - variables globales et variables locales. Selon la classe de la mémoire, elles peuvent être les paramètres d'entrée d'un programme MQL5, statiques et automatiques. Si nécessaire, chaque variable est initialisée par une valeur correspondante. Après avoir été utilisée, une variable est désinitialisée et la mémoire qui était utilisée est rendue au système exécutable MQL5.

Initialisation et Désinitialisation des Variables Globales

Les variables globales sont initialisées automatiquement juste après le chargement d'un programme MQL5 et avant que n'importe quelle fonction ne soit appelée. Pendant l'initialisation, les valeurs initiales sont affectées aux variables de type simple et un constructeur (s'il existe) est appelé pour les objets. Les variables d'entrée sont toujours déclarées au niveau global, et sont initialisées avec les valeurs définies par l'utilisateur dans la boîte de dialogue au lancement du programme.

Bien que les variables statiques sont habituellement déclarées au niveau local, la mémoire pour ces variables est pré-allouée, et l'initialisation est effectuée directement après le chargement du programme, de la même façon que pour les variables globales.

L'ordre d'initialisation correspond à l'ordre de déclaration des variables dans le programme. La désinitialisation est effectuée dans l'ordre inverse. Cette règle n'est vraie que pour les variables qui n'ont pas été créées avec l'opérateur new. Ces variables sont créées et initialisées automatiquement juste après le chargement et sont désinitialisées avant le déchargement du programme.

Initialisation et Désinitialisation des Variables Locales

Si une variable déclarée au niveau local est n'est pas statique, la mémoire est allouée automatiquement pour cette variable. Les variables locales, telles que les variables globales, sont initialisées automatiquement au moment où l'exécution du programme rencontre leur déclaration. L'ordre d'initialisation correspond donc à l'ordre de déclaration.

Les variables locales sont désinitialisées à la fin du bloc du programme dans lequel elles ont été déclarées et dans l'ordre inverse à leur déclaration. Un bloc de programme est un opérateur composé qui peut être une partie d'un opérateur de sélection switch, d'un opérateur de boucle (for, while, do-while), d'un corps de fonction ou d'une partie d'un opérateur if-else.

Les variables ne sont initialisées qu'au moment où l'exécution du programme rencontre la déclaration de la variable. Si pendant l'exécution du programme le bloc dans lequel la variable est déclarée n'est pass exécuté, cette variable n'est pas initialisée.

Initialisation et Désinitialisation des Objets Placés

Les pointeurs d'objet sont un cas spécial, car la déclaration d'un pointeur n'implique pas l'initialisation de l'objet correspondant. Les objets placés dynamiquement ne sont initialisés qu'au moment où l'objet est créé avec l'opérateur new. L'Initialisation des objets présuppose l'appel à un constructeur de la classe correspondante. S'il n'y a pas de constructeur correspondant dans la classe, ses membres de type simple ne seront pas initialisés automatiquement ; ses membres de type string, tableau dynamique et objet complexe seront initialisés automatiquement.

© 2000-2024, MetaQuotes Ltd.

Les pointeurs peuvent être déclarés au niveau local ou global ; et ils peuvent être initialisés avec la valeur vide de NULL oou avec la valeur du pointeur du même type ou du type hérité. Si l'opérateur *new* est appelé pour un pointeur déclaré au niveau local, l'opérateur *delete* pour ce pointeur doit être appelé avant de sortir du niveau. Autrement, le pointeur sera perdu et la suppression explicite de l'objet échouera.

Tous les objets créés par une expression *pointeur_objet*=new *Nom_de_classe*, doivent être supprimés par un opérateur delete(*pointeur_objet*). Si une telle variable n'est pas supprimée pour n'importe quelle raison par l'opérateur delete lorsque le programme est terminé, l'entrée correspondate apparaîtra dans le journal de l'onglet "Experts". Il est possible de déclarer plusieurs variables et de leur assigner à toutes le pointeur d'un même objet.

Si un objet créé dynamiquement a un constructeur, celui-ci sera appelé au moment de l'exécution de l'opérateur *new* . Si un objet a un destructeur, celui-ci sera appelé au moment de l'exécution de l'opérateur *delete*.

Les objets placés dynamiquement ne sont donc créés qu'au moment de l'invocation de leur opérateur *new* , et sont assurés d'être supprimés soit par l'opérateur *delete* ou automatiquement par le système d'exécution de MQL5 lors du déchargement du programme. L'ordre de déclaration des pointeurs des objets créés dynamiquement n'influence pas l'ordre de leur initialisation. L'ordre d'initialisation et de désinitialisation est totalement contrôlé par le programmeur.

Allocation dynamique de la mémoire en MQL5

Lors de l'utilisation de tableaux dynamiques, la mémoire libérée est automatiquement rendue au système d'exploitation.

Lors de l'utilisation d'objets de classe dynamiques avec l'opérateur new, la mémoire est d'abord demandée dans la zone de mémoire de la classe avec laquelle le gestionnaire de mémoire fonctionne. S'il n'y a plus assez de mémoire, la mémoire est demandée au système d'exploitation. Lors de la suppression de l'objet dynamique avec l'opérateur delete, la mémoire libérée est immédiatement rendue à la zone de mémoire de la classe.

Le gestionnaire de mémoire libère la mémoire et la rend au système d'exploitation après la fin des fonctions de gestion des évènements : OnInit(), OnDeinit(), OnStart(), OnTick(), OnCalculate(), OnTimer(), OnTrade(), OnTester(), OnTesterInit(), OnTesterPass(), OnTesterDeinit(), OnChartEvent(), OnBookEvent().

Principales Caractéristiques des Variables

Les principales informations sur l'ordre de création suppression sur les appels des constructeurs et destructeurs sont données dans le tableau ci-dessous.

	Variable automatique globale	Variable automatique locale	Object créé dynamiquement
Initialisation	directement après qu'un programme mql5 soit chargé	lorsque la ligne de code où elle est déclarée est atteinte pendant l'exécution	à l'exécution de l'opérateur **new**

© 2000-2024, MetaQuotes Ltd.

	Variable automatique globale	Variable automatique locale	Object créé dynamiquement
Ordre d'initialisation	dans l'ordre de déclaration	dans l'ordre de déclaration	indépendant de l'ordre de déclaration
Désinitialisation	avant qu'un programme mql5 soit déchargé	lorsque l'exécution sort du bloc de déclaration	lorsque l'opérateur **delete** est exécuté ou avant qu'un programme mql5 soit déchargé
Ordre de désinitialisation	dans l'ordre inverse à l'ordre d'initialisation	dans l'ordre inverse à l'ordre d'initialisation	indépendant de l'ordre d'initialisation
Appel du constructeur	au chargement du programme mql5	à l'initialisation	à l'exécution de l'opérateur *new*
Appel du destructeur	au déchargement du programme mql5	lors de la sortie du bloc où la variable a été initialisée	à l'exécution de l'opérateur *delete*
Journaux des erreurs	écrit un message dans le journal de l'onglet "Experts" sur la tentative de suppression d'un objet créé automatiquement	écrit un message dans le journal de l'onglet "Experts" sur la tentative de suppression d'un objet créé automatiquement	écrit un message dans le journal de l'onglet "Experts" sur la non-suppression d'objets créés dynamiquement au moment du déchargement du programme mql5

Voir aussi

Types de Données, Encapsulation et Extension des Types, Initialisation des Variables, Portée de Visibilité et Cycle de Vie des Variables

© 2000-2024, MetaQuotes Ltd.

Préprocesseur

Le préprocesseur est un sous-système spécial du compilateur MQL5 qui sert à la préparation du code source du programme immédiatement avant que le programme ne soit compilé.

Le préprocesseur permet d'améliorer la lisibilité du code source. Le code peut être structuré en incluant des fichiers spécifiques contenant le code source de programmes mql5. La possibilité d'assigner des noms mnémoniques à des constantes spécifiques permet d'améliorer la lisibilité du code.

Le préprocesseur permet aussi de déterminer les paramètres spécifiques des programmes mql5 :

- Déclaration de constantes
- Définition des propriétés du programme
- Inclusion de fichiers dans le texte du programme
- Import de fonctions
- Compilation Conditionnelle

The preprocessor directives are used by the compiler to preprocess the source code before compiling it. The directive always begins with #, therefore the compiler prohibits using the symbol in names of variables, functions etc.

Each directive is described by a separate entry and is valid until the line break. You cannot use several directives in one entry. If the directive entry is too big, it can be broken into several lines using the '\' symbol. In this case, the next line is considered a continuation of the directive entry.

```
//+------------------------------------------------------------------+
//|  foreach pseudo-operator                                         |
//+------------------------------------------------------------------+
#define ForEach(index, array) for (int index = 0,                    \
   max_##index=ArraySize((array));                                   \
   index<max_##index; index++)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string array[]={"12","23","34","45"};
//--- bypass the array using ForEach
   ForEach(i,array)
     {
      PrintFormat("%d: array[%d]=%s",i,i,array[i]);
     }
  }
//+------------------------------------------------------------------+
/* Output result
   0: array[0]=12
   1: array[1]=23
   2: array[2]=34
   3: array[3]=45
```

© 2000-2024, MetaQuotes Ltd.

```
*/
```

For the compiler, all these three #define directive lines look like a single long line. The example above also applies ## character which is a merge operator used in the #define macros to merge the two macro tokens into one. The tokens merge operator cannot be the first or last one in a macro definition.

© 2000-2024, MetaQuotes Ltd.

Macro de substitution (#define)

The preprocessor directives are used by the compiler to preprocess the source code before compiling it. The directive always begins with #, therefore the compiler prohibits using the symbol in names of variables, functions etc.

Each directive is described by a separate entry and is valid until the line break. You cannot use several directives in one entry. If the directive entry is too big, it can be broken into several lines using the '\' symbol. In this case, the next line is considered a continuation of the directive entry.

La directive #define peut être utilisée pour assigner des noms mnémoniques aux constantes. Il y a deux formes :

```
#define identifiant expression                    // forme sans paramètre
#define identifiant(par1,... par8) expression     // forme paramétrée
```

La directive #define substitue **expression** pour toutes les entrées trouvées de *identifiant* dans le texte source. L'*identifiant* n'est remplacé que s'il est sous forme séparée. L'*identifiant* n'est pas remplacé s'il fait partie d'un commentaire, d'une chaîne de caractères ou d'un autre identifiant plus long.

Les identifiants de constantes obéissent aux mêmes règles que les noms de variables. La valeur peut être de n'importe quel type :

```
#define ABC               100
#define PI                3.14
#define COMPANY_NAME      "MetaQuotes Software Corp."
...
void ShowCopyright()
  {
   Print("Copyright  2001-2009, ",COMPANY_NAME);
   Print("https://www.metaquotes.net");
  }
```

expression peut consister en différents éléments, tels que des mots-clés, des constantes, des expressions constantes et non-constantes. **expression** se termine avec la fin de la ligne et ne peut pas être transférée sur la ligne suivante.

Exemple :

```
#define TWO        2
#define THREE      3
#define INCOMPLETE TWO+THREE
#define COMPLETE   (TWO+THREE)
void OnStart()
  {
   Print("2 + 3*2 = ",INCOMPLETE*2);
   Print("(2 + 3)*2 = ",COMPLETE*2);
  }
// Résultat
// 2 + 3*2 = 8
// (2 + 3)*2 = 10
```

© 2000-2024, MetaQuotes Ltd.

Forme Paramétrique de #define

Avec la forme paramétrique, toutes les entrées trouvées de l'identifiant sont remplacées par l'expression en tenant compte des paramètres donnés. Par exemple :

```
 // exemple avec deux paramètres a et b
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))

double c=MUL(A,B);
Print("c=",c);
/*
expression double c=MUL(A,B);
est équivalent à double c=((2+3)*(5-1));
*/
// Résultat
// c=20
```

Assurez-vous d'entourer les paramètres entre parenthèses lors de l'utilisation des paramètres dans expression, car ceci aidera à éviter des erreurs non évidentes qui sont difficiles à trouver. Si le code est réécrit sans les parenthèses, le résultat sera différent :

```
 // exemple avec deux paramètres a et b
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
expression double c=MUL(A,B);
est équivalent à double c=2+3*5-1;
*/
// Résultat
// c=16
```

Lors de l'utilisation de la forme paramétrique, un maximum de 8 paramètres est autorisé.

```
 // forme paramétrique correcte
#define LOG(text)  Print(__FILE__,"(",__LINE__,") :",text)   // un paramètre - 'text'

 // forme paramétrique incorrecte
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9)   p1+p2+p3+p4 // plus de 8 paran
```

La directive #undef

© 2000-2024, MetaQuotes Ltd.

La directive #undef annule la déclaration de la substitution de macro définie auparavant.

Exemple :

```
#define MACRO

void func1()
  {
#ifdef MACRO
   Print("MACRO est définie dans ",__FUNCTION__);
#else
   Print("MACRO n'est pas définie dans ",__FUNCTION__);
#endif
  }

#undef MACRO

void func2()
  {
#ifdef MACRO
   Print("MACRO est définie dans ",__FUNCTION__);
#else
   Print("MACRO n'est pas définie dans ",__FUNCTION__);
#endif
  }

void OnStart()
  {
   func1();
   func2();
  }

/* Résultat :
MACRO est définie dans func1
MACRO n'est pas définie dans func2
*/
```

Voir aussi

Identifiants, Constantes Caractères

© 2000-2024, MetaQuotes Ltd.

Propriétés du Programme (#property)

Chaque programme mql5 permet de spécifier des paramètres spécifiques supplémentaires préfixés #property qui aident le terminal client à présenter le programme sans avoir besoin de le lancer explicitement. Ceci concerne avant tout les paramètres externes des indicateurs. Les propriétés décrites dans les fichiers inclus sont complètement ignorées. Les propriétés doivent être spécifiées dans le principal fichier mq5.

```
#property identifiant valeur
```

Le compilateur écrira les valeurs déclarées à la configuration du module exécuté.

Constante	Type	Description
icone	string	Chemin vers le fichier d'une image qui sera utilisée comme icone du programme EX5. Les règles de spécification du chemin sont les mêmes que pour les resources. La propriété doit être spécifiée dans le module principal avec le code source MQL5. Le fichier de l'icone doit être au format ICO.
link	string	Lien vers le site Internet de la société
copyright	string	Le nom de la société
version	string	Version du programme, maximum 31 caractères
description	string	Description textuelle brève du programme mql5. Plusieurs *descriptions* peuvent être présentes, chacune d'elle décrit une ligne de texte. La longueur totale de toutes les *descriptions* ne doit pas être supérieure à 511 caractères, incluant les retours chariot.
stacksize	int	Taille de la pile du programme MQL5. Une pile de taille suffisante est nécessaire lors de l'exécution d'appels récursifs de fonctions. Lors du lancement d'un script ou d'un Expert Advisor sur le graphique, une pile d'au moins 8 Mo est allouée. Dans le cas des indicateurs, la taille de la pile est toujours fixée et égale à 1 Mo. Lorsqu'un programme est lancé dans le strategy tester, une pile de 16 Mo est toujours allouée.
library		Une bibliothèque de code ; aucune fonction de lancement n'est assignée, les fonctions ayant le modificateur export peuvent être importées dans d'autres programmes mql5
indicator_applied_price	int	Spécifie la valeur par défaut du champ "Appliqué à". Vous pouvez spécifier l'une des valeurs de ENUM_APPLIED_PRICE. Si la propriété n'est pas configurée, la valeur par défaut est PRICE_CLOSE

© 2000-2024, MetaQuotes Ltd.

Constante	Type	Description
indicator_chart_window		Affiche l'indicateur dans la fenêtre principale du graphique
indicator_separate_window		Affiche l'indicateur dans une fenêtre séparée
indicator_height	int	Hauteur fixe de la sous-fenêtre de l'indicateur, en pixels (propriété INDICATOR_HEIGHT)
indicator_buffers	int	Nombre de buffers pour les calculs de l'indicateur
indicator_plots	int	Nombre de séries graphiques dans l'indicateur
indicator_minimum	double	La valeur limite basse pour la fenêtre séparée de l'indicateur
indicator_maximum	double	La valeur limite haute pour la fenêtre séparée de l'indicateur
indicator_labelN	string	Définit le label pour la N-ième série graphique affichée dans la Fenêtre des Données Pour les séries graphiques nécessitant plusieurs buffers (DRAW_CANDLES, DRAW_FILLING et autres), les noms des étiquettes sont définis en utilisant le séparateur ';'.
indicator_colorN	color	La couleur d'affichage de la N-ième ligne, où N est le numéro de la série graphique ; la numérotation commence à 1
indicator_widthN	int	L'épaisseur de la N-ième ligne, où N est le numéro de la série graphique ; la numérotation commence à 1
indicator_styleN	int	Le style de ligne de la série graphique, spécifié par les valeurs de ENUM_LINE_STYLE. N est le numéro de la série graphique ; la numérotation commence à 1
indicator_typeN	int	Le type du dessin graphique, spécifié par les valeurs de ENUM_DRAW_TYPE. N est le numéro de la série graphique ; la numérotation commence à 1
indicator_levelN	double	Valeur du N-ième niveau horizontal dans la fenêtre séparée d'un indicateur
indicator_levelcolor	color	Couleur des niveaux horizontaux de l'indicateur
indicator_levelwidth	int	Epaisseur des niveaux horizontaux de l'indicateur
indicator_levelstyle	int	Style des niveaux horizontaux de l'indicateur
script_show_confirm		Affiche une fenêtre de confirmation avant de lancer le script

© 2000-2024, MetaQuotes Ltd.

Constante	Type	Description
script_show_inputs		Affiche la fenêtre des propriétés avant d'exécuter le script et désactive la fenêtre de confirmation
tester_indicator	string	Nom d'un indicateur personnalisé au format "*nom_indicateur.ex5*". Les indicateurs nécessitant d'être testés sont définis automatiquement à partir de l'appel de la fonction iCustom(), si le paramètre correspondant est défini par une chaîne de caractères constante. Pour tous les autres cas (utilisation de la fonction IndicatorCreate() ou utilisation d'une chaîne de caractères non constante dans le paramètre définissant le nom de l'indicateur), cette propriété est nécessaire
tester_file	string	Le nom du fichier d'un testeur avec son extension, entre guillemets (comme une chaîne de caractères constante). Le fichier spécifié sera passé au testeur. Les fichiers d'entrée à tester, s'il y en a, doivent toujours être spécifiés.
tester_library	string	Le nom d'une bibliothèque de code avec son extension, entre guillemets. Une bibliothèque peut avoir une extension de type 'dll' ou 'ex5'. Les bibliothèques qui doivent être testées sont définies automatiquement. Cependant, si l'une des bibliothèques est utilisée par un indicateur personnalisé, cette propriété est requise
tester_set	string	Le nom fichier .set avec les valeurs des paramètres d'entrée. Le fichier est passé au testeur avec le test et l'optimlisation. Ce nom de fichier est spécifié avec l'extension et entre guillemets sous la forme d'une chaîne de caractères constante. Si vous spécifiez le nom de l'EA et le numéro de la version sous la forme "<nom_expert>_<version>.set" dans le nom du fichier, il est alors automatiquement ajouté au menu de téléchargement des versions du paramètre. Par exemple, le nom "MACD Sample_4.set" signifie que c'est un fichier de valeurs de paramètres pour l'EA "MACD Sample.mq5" et la version 4. Pour définir le format, nous vous recommandons de sauvegarder manuellement les paramètres de test/optimisation dans le testeur de stratégie et d'ouvrir ensuite le fichier .set créé de cette façon.

© 2000-2024, MetaQuotes Ltd.

Constante	Type	Description
tester_no_cache	string	Lorsque vous effectuez une optimisation, le strategy tester sauvegarde tous les résultats des passes exécutées dans le cache d'optimisation, dans lequel le résultat du test est sauvegardé pour chaque ensemble de paramètres d'entrée. Ceci permet d'utiliser les résultats déjà prêts lors d'une seconde optimisation des mêmes paramètres sans perdre de temps à recalculer. Mais dans certaines tâches (par exemple des calculs mathématiques), il peut être nécessaire de refaire les calculs indépendamment de la disponibilité de résultats déjà effectués dans le cache d'optimisation. Dans ce cas, le fichier devrait inclure la propriété *tester_no_cache*. Les résultats de test sont toujours stockés dans le cache, vous pouvez donc voir toutes les données des passes effectuées dans le strategy tester.
tester_everytick_calculate	string	Dans le Strategy Tester, les indicateurs ne sont calculés que lorsque vous accédez à ses données. Ceci accélère grandement la vitesse du test et de l'optimisation, si vous n'avez pas besoin d'obtenir les valeurs de l'indicateur à chaque tick. En spécifiant la propriété *tester_everytick_calculate*, vous pouvez activer le calcul forcé de l'indicateur sur chaque tick. Les calculs des indicateurs dans le Strategy Tester sont également forcés sur chaque tick dans les cas suivants : • lors du test en mode visuel ; • si l'indicateur a l'une des fonctions suivantes : EventChartCustom, OnChartEvent, OnTimer; • si l'indicateur a été créé avec un compilateur dont le numéro de build est inférieur à 1916. Cette fonctionnalité ne s'applique qu'au Strategy Tester, les indicateurs du terminal sont toujours calculés sur chaque tick reçu.
optimization_chart_mode	string	Spécifie le type du graphique et les noms de 2 paramètres d'entrée qui seront utilisés pour la visualisation des résultats de l'optimisation. Par exemple, "3d, InpX, InpY" signifie que les résultats seront affichés en 3D avec les axes des

© 2000-2024, MetaQuotes Ltd.

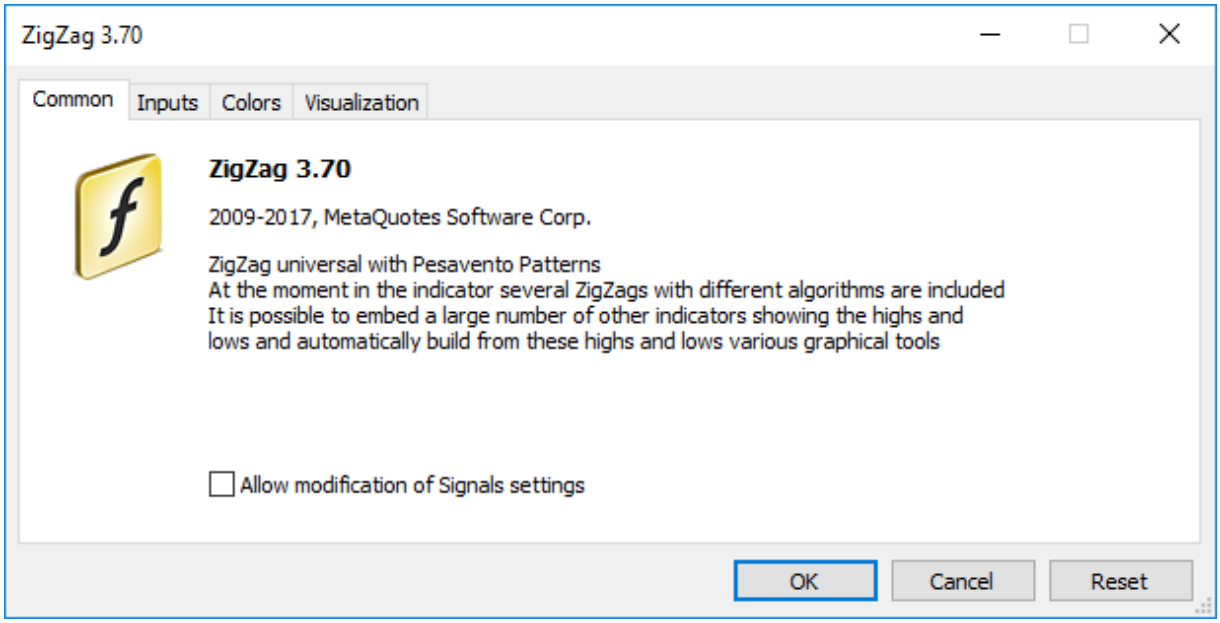
Constante	Type	Description
		coordonnées basés sur les valeurs des paramètres InpX et InpY. La propriété permet donc d'activer la spécification des paramètres qui seront utilisés pour afficher le graphique d'optimisation et le type du graphique directement dans le code du programme. Options possibles : • "3d, input_parameter_name1, input_parameter_name2" signifie un graphique de visualisation en 3D, sur lequel il est possible de tourner, de zoomer et de dézoomer. Le graphique est construit en utilisant 2 paramètres. • "2d, input_parameter_name1, input_parameter_name2" signifie un graphique en 2D, dans lequel chaque cellule sera d'une certaine couleur, suivant le résultat. Le graphique est construit en utilisant 2 paramètres. • "1d, input_parameter_name1, input_parameter_name2" signifie un graphique en courbe, dans lequel les résultats sont triés suivant le paramètre spécifié. Chaque passe est affichée sous la forme d'un point. Le graphique est construit en utilisant 1 seul paramètre. • "0d, input_parameter_name1, input_parameter_name2" signifie un graphiqie normal avec les résultats triés par l'heure d'arrivée du résultat de la passe. Chaque passe est affichée sous la forme d'un point dans le graphique. L'indication du paramètre n'est pas obligatoire, mais les paramètres spécifiés peuvent être utilisés pour basculer manuellement vers d'autres types de graphiques. Vous ne pouvez indiquer également que le type du graphique, sans spécifier de paramètre d'entrée. Dans ce cas, le terminal sélectionnera les paramètres nécessaires pour afficher le graphique d'optimisation.

Exemple de Description et de Numéro de Version

```
#property version     "3.70"      // Version actuelle de l'Expert Advisor
#property description "ZigZag universal with Pesavento Patterns"
#property description "At the moment in the indicator several ZigZags with different a
#property description "It is possible to embed a large number of other indicators show
```

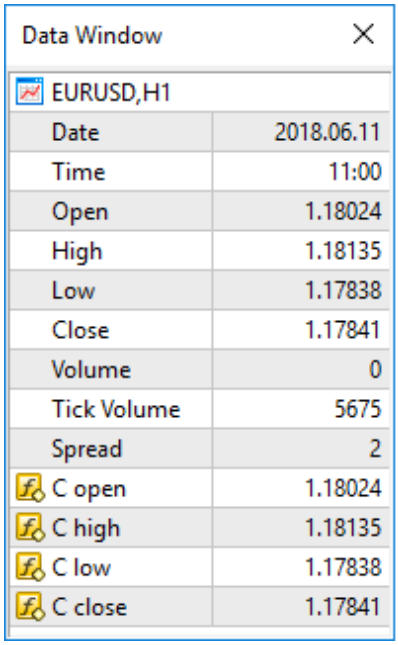
© 2000-2024, MetaQuotes Ltd.

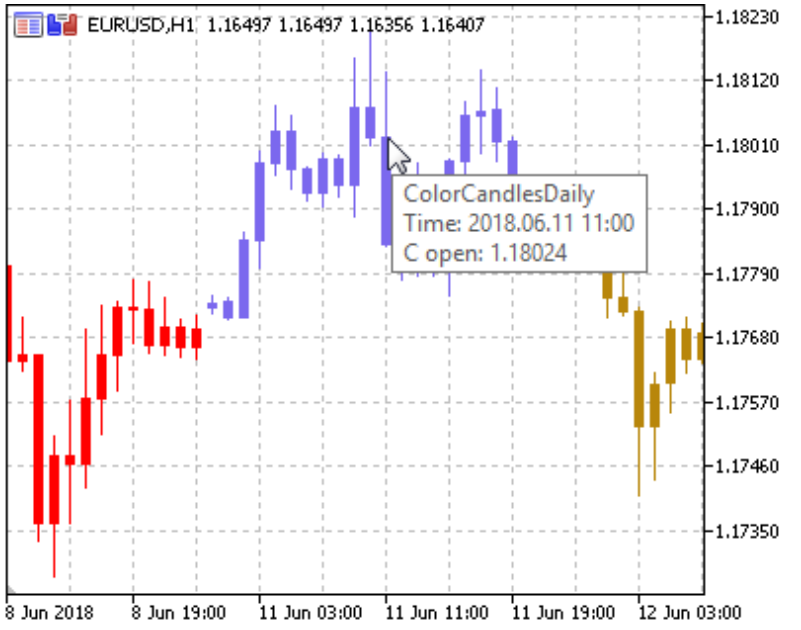
```
#property description "lows and automatically build from these highs and lows various
```



Exemples de Sépcification d'un Label Séparé pour Chaque Buffer de l'Indicateur ("C open; C high; C low; C close")

```
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_plots   1
#property indicator_type1   DRAW_CANDLES
#property indicator_width1  3
#property indicator_label1  "C open;C high;C low;C close"
```





© 2000-2024, MetaQuotes Ltd.

Inclure des Fichiers (#include)

La ligne de commande *#include* peut être placée n'importe où dans le programme, mais habituellement, toutes les inclusions sont placées au début du code source. Format d'appel :

```
#include <nom_fichier>
#include "nom_fichier"
```

Exemples :

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

Le préprocesseur remplace la ligne *#include <nom_fichier>* avec le contenu du fichier WinUser32.mqh. Les crochets indiquent que le fichier WinUser32.mqh sera pris depuis le répertoire standard (habituellement *répertoire_installation_terminal\MQL5\Include*). Le répertoire courant n'est pas inclus dans la recherche.

Si le nom du fichier est entre guillements, la recherche est effectuée dans le répertoire courante (qui contient le fichier source principal). Le répertoire standard n'est pas inclus dans la recherche.

Voir aussi

Bibliothèque Standard, Importer des Fonctions

© 2000-2024, MetaQuotes Ltd.

Importer une Fonction (#import)

Les fonctions sont importées depuis des modules MQL5 compilés (fichiers *.ex5) et des modules du système d'exploitation (fichiers *.dll). Le nom du module est spécifié dans la directive *#import*. Pour que le compilateur puisse former correctement l'appel à la fonction importée et organiser la transmission des paramètres correctement, la description complète des fonctions est nécessaire. Les descriptions des fonctions suivent immédiatement la directive *#import "nom_module"*. La nouvelle commande *#import* (peut être sans paramètres) termine le bloc des descriptions des fonctions importées.

```
#import "nom_du_fichier"
    func1 define;
    func2 define;
    ...
    funcN define;
#import
```

Les fonctions importées peuvent avoir plusieurs noms. Les fonctions ayant le même nom mais des modules différents peuvent être importés en même temps. Les fonctions importées peuvent avoir des noms qui correspondent aux noms des fonctions intégrées. L'opération de résolution de portée définit quelles fonctions doivent être appelées.

L'ordre de recherche d'un fichier spécifié après le mot-clé #import est décrit dans Appel des Fonction Importées.

Puisque les fonctions importées sont en dehors du module compilé, le compilateur ne peut pas vérifier la validité des paramètres passés. Donc, pour éviter des erreurs d'exécution (runtime), la composition et l'ordre des paramètres passés aux fonctions importés doivent être décrits de façon précise. Les paramètres passés aux fonctions importées (à la fois des modules EX5 et DLL) peuvent avoir des valeurs par défaut.

Les suivants ne peuvent pas être utilisés comme paramètres des fonctions importées :

- pointeurs (*);
- liens vers des objets contenant des tableaux dynamiques et/ou des pointeurs.

Les classes, tableaux de chaînes de caractères ou d'objets complexes qui contiennent des chaînes de caractères et/ou des tableaux dynamiques de n'importe quels types ne peuvent pas être passés comme paramètres aux fonctions importées des DLL.

Exemples :

```
#import "stdlib.ex5"
string ErrorDescription(int error_code);
int    RGB(int red_value,int green_value,int blue_value);
bool   CompareDoubles(double number1,double number2);
string DoubleToStrMorePrecision(double number,int precision);
string IntegerToHexString(int integer_number);
#import "ExpertSample.dll"
int    GetIntValue(int);
double GetDoubleValue(double);
string GetStringValue(string);
double GetArrayItemValue(double &arr[],int,int);
```

© 2000-2024, MetaQuotes Ltd.

```
bool   SetArrayItemValue(double &arr[],int,int,double);
double GetRatesItemValue(double &rates[][6],int,int,int);
#import
```

Pour importer des fonctions pendant l'exécution d'un programme mql5, la liaison anticipée (early binding) est utilisée. Cela signifie que la bibliothèque est chargée pendant le chargement d'un programme utilisant son programme ex5.

Il n'est pas recommandé d'utiliser le nom qualifié complet (fully qualified name) du module chargeable du type *Lecteur:\Répertoire\NomFichier.Ext*. Les bibliothèques MQL5 sont chargées à partir du répertoire *rep_terminal\MQL5\Libraries*.

Si la fonction importée a des versions d'appel différentes pour les versions de Windows 32 et 64 bits, les deux doivent être importées et la bonne version de la fonction sera appelée explicitement avec la variable _IsX64.

Exemple :

```
#import "user32.dll"
//--- Pour le système 32 bits
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
//--- Pour le système 64 bits
int    MessageBoxW(ulong hWnd,string lpText,string lpCaption,uint uType);
#import
//+------------------------------------------------------------------+
//|  MessageBox_32_64_bit utilise la bonne version de MessageBoxW()  |
//+------------------------------------------------------------------+
int MessageBox_32_64_bit()
  {
   int res=-1;
   //--- Si nous utilisons Windows 64 bits
   if(_IsX64)
     {
      ulong hwnd=0;
      res=MessageBoxW(hwnd,"Exemple d'appel à MessageBoxW 64 bits","MessageBoxW 64 bit
     }
   else  // Nous utilisons Windows 32 bits
     {
      uint hwnd=0;
      res=MessageBoxW(hwnd,"Exemple d'appel à MessageBoxW 32 bits","MessageBoxW 32 bit
     }
   return (res);
  }
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int ans=MessageBox_32_64_bit();
```

© 2000-2024, MetaQuotes Ltd.

```
   PrintFormat("MessageBox_32_64_bit a retourné %d",ans);
  }
```

Importer des fonctions de bibliothèques .NET

Pour utiliser les fonctions des bibliothèques .NET, importez simplement la DLL sans définir de fonctions spécifiques. MetaEditor importe automatiquement toutes les fonctions utilisables :

- Structures simples (POD, plain old data) — structures qui ne contiennent que des types de données simples.
- Fonctions statiques publiques ayant des paramètres, dans lesquelles seuls des types simples et des structures POD, ou leurs tableaux, sont utilisés.

Pour appeler des fonctions de la bibliothèque, il suffit de l'importer :

```
#import "TestLib.dll"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int x=41;
   TestClass::Inc(x);
   Print(x);
  }
```

Le code C# de la fonction Inc de TestClass ressemble à:

```
public class TestClass
  {
   public static void Inc(ref int x)
     {
      x++;
     }
  }
```

Le script retourne la valeur 42 comme résultat de l'exécution.

Voir aussi

Inclure des Fichiers

© 2000-2024, MetaQuotes Ltd.

Compilation Conditionnelle (#ifdef, #ifndef, #else, #endif)

The preprocessor directives are used by the compiler to preprocess the source code before compiling it. The directive always begins with #, therefore the compiler prohibits using the symbol in names of variables, functions etc.

Each directive is described by a separate entry and is valid until the line break. You cannot use several directives in one entry. If the directive entry is too big, it can be broken into several lines using the '\' symbol. In this case, the next line is considered a continuation of the directive entry.

Les directives de compilation conditionnelle du préprocesseur permettent de compiler ou d'ignorer une partie du programme suivant une certaine condition.

Cette condition peut prendre l'une des formes suivantes.

```
#ifdef identifiant
   // le code situé ici est compilé si l'identifiant a été défini dans une directive #
#endif
```

```
#ifndef identifiant
   // le code situé ici est compilé si l'identifiant n'est pas défini dans une directi
#endif
```

Toutes les directives de compilation conditionnelles peuvent être suivis par un certain nombre de lignes contenant peut être une directive #else et se terminant par #endif. Si la condition vérifiée est vraie, les lignes entre #else et #endif sont ignorées. Si la condition vérifiée est fausse, toutes les lignes entre les directives #ifdef/#ifndef et #else (ou #endif s'il n'y a pas de #else) sont ignorées.

Exemple :

```
#ifndef TestMode
   #define TestMode
#endif
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef TestMode
      Print("Mode de test");
   #else
      Print("Mode normal");
   #endif
  }
```

Suivant le type du programme et le mode de compilation, les macros standards sont définies de la façon suivante :

La macro __MQL5__ est définie lors de la compilation d'un fichier *.mq5, la macro __MQL4__ est définie lors de la compilation d'un fichier *.mq4.
La macro _DEBUG est définie lors d'une compilation en mode debug.
La macro _RELEASE est définie lors d'une compilation en mode release.

© 2000-2024, MetaQuotes Ltd.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef __MQL5__
      #ifdef _DEBUG
         Print("Hello de la part du compilateur MQL5 [DEBUG]");
      #else
        #ifdef _RELEASE
           Print("Hello de la part du compilateur MQL5 [RELEASE]");
        #endif
     #endif
   #else
      #ifdef __MQL4__
         #ifdef _DEBUG
           Print("Hello de la part du compilateur MQL4 [DEBUG]");
        #else
           #ifdef _RELEASE
              Print("Hello de la part du compilateur MQL4 [RELEASE]");
           #endif
        #endif
     #endif
   #endif
  }
```

© 2000-2024, MetaQuotes Ltd.

Programmation Orientée Objet

La Programmation Orientée-Objet (POO ou OOP en anglais) est la programmation concentrée sur les données, les données et les comportements sont inséparables. Les données et les comportements ensemble constituent une classe, tandis que les objets sont des instances de classe.

Les composantes de l'approche orientée-objet sont :

- Encapsulation et extensibilité de type
- Héritage
- Polymorphisme
- Surcharge
- Fonctions virtuelles

La POO considère les calculs comme la modélisation des comportements. L'élément modélisé est l'objet représneté par des abstractions de calcul. Supposons que nous voulions écrire le fameux jeu "Tetris". Pour ce faire, nous devons savoir comment modéliser l'apparence des formes aléatoires composées de quatre carrés reliés ensemble par les bords. Nous devons également régler la vitesse de chute des formes, définir les opérations de rotation et de décalage des formes. Le déplacement des formes à l'écran est limité par les bordures du puits, cette exigence doit également être modélisée. De plus, les rangées pleines de cubes doivent être détruites et le nombre de points doit être compté.

Ce jeu facile à comprendre nécessite donc la création de plusieurs modèles - modèle de la forme, modèle du puits, modèle du mouvement des formes, etc. Tous ces modèles sont des abstractions, représentées par des calculs de l'ordinateur. Pour décrire ces modèles, le concept de Type de Données Abstrait, ADT (ou type de donnée complexe) est utilisé. A propement parler, le modèle de mouvement des "formes" dans le DOM n'est pas un type de données, mais est un ensemble d'opérations du type de données "forme", utilisant les restrictions du type de données "puits".

Les objets sont des variables de classe. La programmation orientée-objet vous permet de créer ADT et de l'utiliser facilement. La programmation orientée-objet utilise le mécanisme d'héritage. L'héritage permet d'obtenir des types dérivés des types de données déjà définis par l'utilisateur.

Par exemple, pour créer les formes du Tetris, il est plus pratique de créer en premier une classe de base Shape. Les autres classes représentant les 7 autres types possibles de formes peuvent dériver de cette base. Le comportement des formes est défini dans la classe de base, tandis l'implémentation des comportements de chaque forme est défini dans les classes dérivées.

En POO, les objets sont responsables de leur comportement. Un développeur ADT devrait inclure du code pour décrire n'importe quel comportement qui serait attendu des objets correspondants. Le fait que l'objet lui-même soit responsable de son comportement simplifie grandement la programmation pour l'utilisateur de cet objet.

Si nous souhaitons dessiner une forme à l'écran, nous devons savoir où se trouvera le centre et comment le dessiner. Si une autre forme sait comment se dessiner elle-même, le programmeur ne doit pas envoyer un message "draw" lorsqu'il utilise cette forme.

Le langage MQL5 ressemble au C++, et possède également le mécanisme d'encapsulation por l'implémentation d'ADT. D'un côté, l'encapsulation intègre les détails internes de l'implémentation d'un type particulier, et de l'autre, elle regroupe les fonctions accessibles de l'extérieur qui peuvent agir sur les objets de ce type. Les détails d'implémentation peuvent ne pas être accessibles pour un programme utilisant ce type.

© 2000-2024, MetaQuotes Ltd.

Le concept de POO a un ensemble de concepts liés, incluant les suivants :

- Simulation d'actions du monde réel
- Types de données définis par l'utilisateur
- Masquage des détails d'implémentation
- Possibilité de réutilisation du code grâce à l'héritage
- Interpretation des appels de fonctions pendant l'exécution

Certains de ces concepts sont assez vagues, certains sont abstraits, d'autres plus généraux.

© 2000-2024, MetaQuotes Ltd.

Encapsulation et Extensibilité des Types

La POO est une approche équilibrée pour écrire des logiciels. Les données et les comportements sont regroupés ensemble. Cette encapsulation crée des types de données définis par l'utilisateur, étendant les types de données du langage et les utilisant. L'extensibilité des types est une opportunité pour ajouter au langage des types de données définis par l'utilisateur, qui sont également faciles à utiliser, ainsi que des types de base.

Un type de données abstrait, par exemple, une chaîne de caractères, est la description d'un type de comportement idéal et bien connu.

L'utilisateur de la chaîne de caractères sait que les opérations sur les chaînes, telles que la concaténation ou l'affichage, ont un comportement spécifique. Les opérations de concaténation et d'affichage sont appelées des méthodes.

Une certaine implémentation d'ADT peut avoir quelques restrictions, par exemple, les chaînes de caractères peuvent avoir une longueur maximum. Ces limitations affectent le comportement ouvert à tous. En même temps, les détails internes ou privés d'implémentation n'affectent pas directement la façon dont l'utilisateur voit l'objet. Par exemple, la chaîne de caractères est souvent implémentée sous la forme d'un tableau, tandis que l'adresse de base interne de ce tableau et son nom ne sont pas essentiels.

L'encapsulation est la possibilité de cacher les détails d'implémentation lorsque les interfaces du type défini par l'utilisateur sont fournis. En MQL5, comme en C++, les définitions des classes et des structures (class et struct) sont utilisées en prévision de l'encapsulation en combinaison avec les mot-clés d'accès private, protected et public.

Le mot-clé public indique que l'accès aux membres situés après ce mot est ouvert sans restrictions. Sans ce mot-clé, les membres de classe sont inacessibles par défaut. Les membres privés ne sont accessibles que par les fonctios membres de cette classe.

Les fonctions protected de la classe sont disponibles aux fonctions non seulement de cette classe, mais aussi des classes héritées. Les fonctions publiques de la classe sont disponibles pour toutes les fonctions dans la portée de la déclaration de la classe. Cette protection rend possible de masquer des parties de l'implémentation de la classe, donc de prévenir tout changement inattendu dans la structure des données. La restriction d'accès ou masquage des données est une caractéristique de la programmation orientée-objet.

Habituellement, les fonctions de classe sont protégées et déclarées avec le mot-clé protected , la lecture et l'écriture des valeurs sont effectuées par des méthodes spécifiques set- et get- qui sont définies avec le spécificateur d'accès public.

Exemple :

```
class CPerson
  {
protected:
   string            m_name;                    // nom
public:
   void              SetName(string n){m_name=n;}// définit le nom
```

© 2000-2024, MetaQuotes Ltd.

```
   string            GetName(){return (m_name);} // retourne le nom
  };
```

Cette approche offre plusieurs avantages. Premièrement, il est possible de comprendre ce que fait une fonction grâce à son nom - définit (set) ou retourne (get) la valeur d'un membre de la classe. Deuxièmement, il est possible de changer par la suite le type de la variable m_name de la classe CPerson ou de l'une de ses classes dérivées.

Dans ce cas, nous n'aurons qu'à changer l'implémentation des fonctions SetName() et GetName(), tandis que les objets de la classe CPerson pourront être utilisés dans un programme sans changer de code car l'utilisateur ne saura même pas que le type de données de m_name a changé.

Exemple :

```
struct Name
  {
   string            first_name;                 // prénom
   string            last_name;                  // nom
  };

class CPerson
  {
protected:
   Name              m_name;                     // nom
public:
   void              SetName(string n);
   string            GetName(){return(m_name.first_name+" "+m_name.last_name);}
private:
   string            GetFirstName(string full_name);
   string            GetLastName(string full_name);
  };

void CPerson::SetName(string n)
  {
   m_name.first_name=GetFirstName(n);
   m_name.last_name=GetLastName(n);
  }

string CPerson::GetFirstName(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>0) StringSetCharacter(full_name,pos,0);
   return(full_name);
  }

string CPerson::GetLastName(string full_name)
  {
   string ret_string;
```

© 2000-2024, MetaQuotes Ltd.

```
  int pos=StringFind(full_name," ");
  if(pos>0) ret_string=StringSubstr(full_name,pos+1);
  else      ret_string=full_name;
  return(ret_string);
 }
```

Voir aussi

Types de Données

© 2000-2024, MetaQuotes Ltd.

Héritage

La principale caractéristique de la POO est d'encourager la réutilisation de code au travers de l'héritage. Une nouvelle classe est créée à partir d'une classe existante, appelée classe de base. La classe dérivée utilise les membres de la classe de base, mais peut également les modifier ou en ajouter.

De nombreux types sont des variations de types existants. Il est souvent fastidieux de développer du nouveau code pour chacun d'entre eux. De plus, du nouveau code signifie aussi de nouvelles erreurs. La classe dérivé hérite de la description de la classe de base, donc tout re-développement et re-test du code n'est pas nécessaire. Les relations d'héritage sont hiérarchiques.

La hiérarchie est la méthode qui permet de copier les éléments dans leur diversité et dans leur complexité. Elle introduit la classification des objets. Par exemple, la table périodique des éléments contient des gza. Ils possèdent les propriétés inhérente à tout élément périodique.

Les gaz inertes constituent la sous-classe importante suivante. La hiérarchie est qu'un gaz inerte, tel qu'argon, est un gaz, et un gaz à son tour est une partie du système. Une telle hériarchie permet d'interpréter le comportement des gaz inertes facilement. Nous savons que leurs atomes contiennent des protons et des électrons, ce qui est vrai pour tous les autres éléments.

Nous savons qu'il sont à l'état gazeux à température ambiante, comme tous les gaz. Nous savons qu'aucun gaz de la sous-classe des gaz inertes, n'entre en réaction chimique avec d'autres éléments, et c'est une propriété de tous les gaz inertes.

Considérons l'exemple de l'héritage des formes géométriques. Pour décrire toute la variété des formes simples (cercle, triangle, rectangle, carré, etc.), la meilleure façon est de créer une classe de base (ADT), qui est l'ancêtre de toutes les classes dérivées.

Créons une classe de base CShape, qui ne contient que les membres les plus communs décrivant une forme. Ces membres décrivent les propriétés qui sont caractéristiques de n'importe quelle forme - le type de la forme et les coordonnées de son principal point d'ancrage.

Exemple :

```
//--- La classe de base Shape
class CShape
  {
protected:
   int       m_type;                    // Type de la forme
   int       m_xpos;                    // X - coordonnées du point de base
   int       m_ypos;                    // Y - coordonnées du point de base
public:
             CShape(){m_type=0; m_xpos=0; m_ypos=0;} // constructeur
   void      SetXPos(int x){m_xpos=x;} // assigne X
   void      SetYPos(int y){m_ypos=y;} // assigne Y
  };
```

Ensuite, créons de nouvelles classes dérivées de la classe de base dans laquelle nous ajouterons les champs nécessaires, chacun spécifiant une certaine classe. Pour la forme Cercle, il est nécessaire d'ajouter un membre qui contient la valeur du rayon. La forme Carré est caractérisée par la valeur du côté. Donc, les classes dérivées, héritées de la classe de base CShape seront déclarées comme suit :

© 2000-2024, MetaQuotes Ltd.

```
//--- La classe dérivée cercle
class CCircle : public CShape        // Après le symbole ':', nous spécifions la class
  {                                    // à partir de laquelle l'héritage est fait
private:
   int             m_radius;           // rayon du cercle

public:
                   CCircle(){m_type=1;}// constructeur, type 1
  };
```

Pour la forme Carré, la déclaration de la classe est similaire :

```
//--- la classe dérivée Carré
class CSquare : public CShape          // Après le symbole ':', nous spécifions la cla
  {                                    // à partir de laquelle l'héritage est fait
private:
   int            m_square_side;       // côté du carré

public:
                  CSquare(){m_type=2;} // constructeur, type 2
  };
```

Il est à noter qu'au moment de la création de l'objet, la constructeur de la classe de base est appelé en premier, et le constructeur de la classe dérivée est ensuite appelée. Lorsqu'un objet est détruit, le destructeur de la classe dérivée est appelé en premier, et ensuite le destructeur de la classe de base est appelé.

En déclarant les membres les plus généraux dans la classe de base, nous pouvons donc ajouter des membres supplémentaires dans les classes dérivées pour spécifier des classes particulières. L'héritage permet de créer de puissantes bibliothèques de code qui peuvent être réutilisées plusieurs fois.

La syntaxe pour créer une classe dérivée à partir d'une classe existante est de la forme suivante :

```
class class_name :
          (public | protected | private) opt  base_class_name
  {
   déclaration des membres de la classe
  };
```

L'un des aspects de la classe dérivée est la visibilité (ouverture) de ses membres successeurs (héritiers). Les mots-clés public, protected et private sont utilisés pour indiquer la mesure dans laquelle les membres de la classe de base seront disponibles pour les classes dérivées. le mot-clé public après le symbole ':' dans l'en-tête de la classe dérivée indique que les membres protected et public de la classe de base CShape doivent être hérités comme membres protected et public members dans la classe dérivée CCircle.

Les membres privés de la classe de base ne sont pas disponibles pour la classe dérivée. L'héritage public signifie égaement que les classes dérivées (CCircle et CSquare) sont des CShapes. Le Carré (CSquare) est une forme (CShape), mais la forme n'est pas forcément un carré.

© 2000-2024, MetaQuotes Ltd.

La classe dérivée est une modification de la classe de base, elle hérite des membres protected et public de la classe de base. Les constructeurs et destructeurs de la classe de base ne peuvent pas être hérités. En plus des membres de la classe de base, de nouveaux membres sont ajoutés dans une classe dérivée.

La classe dérivée peut inclure l'implémentation des fonctions membres, différente de la classe de base. Cela n'a rien à voir avec une surcharge, lorsque la signification du même nom de fonction peut être différente pour différentes signatures.

Dans l'héritage protégé, les membres public et protected de la classe de base deviennent des membres protected de la classe dérivée. Dans l'héritage privé, les membres public et protected de la classe de base deviennent des membres privés de la classe dérivée.

Dans l'héritage protégé et privé, la relation comme quoi "l'objet d'une classe dérivée est un objet de la classe de base" n'est pas vraie. Les types d'héritage protégé et privé sont rares, et chacun d'eux doit être utilisé avec précaution.

Il faut comprendre que le type de l'héritage (public, protected ou private) n'affecte pas la façon d'**accéder aux membres des classes de base dans la hiérarchie d'héritage depuis une classe dérivée**. Avec n'importe quel type d'héritage, seuls les membres de la classe de base déclarés avec les spécificateurs public et protected seront disponibles en dehors des classes dérivées. Considérons cela dans l'exemple suivant :

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| Classe d'exemple avec plusieurs types d'accès                    |
//+------------------------------------------------------------------+
class CBaseClass
  {
private:             //--- Le membre privé n'est pas disponible depuis les classes dér
   int               m_member;
protected:           //--- La méthode protected est disponible depuis la classe de bas
   int               Member(){return(m_member);}
public:              //--- Le constructeur de la classe est disponible pour tous les r
                     CBaseClass(){m_member=5;return;};
private:             //--- Une méthode private pour affecter une valeur à m_member
   void              Member(int value) { m_member=value;};

  };
//+------------------------------------------------------------------+
//| Classe dérivée avec des erreurs                                  |
//+------------------------------------------------------------------+
class CDerived: public CBaseClass // la spécification d'héritage public peut être manq
  {
public:
   void Func() // Dans la classe dérivée, définit une fonction avec des appels aux mer
     {
      //--- Tentative de modifier un membre privé de la classe de base
      m_member=0;        // Erreur, le membre privé de la classe de base n'est pas dis
```

```
      Member(0);          // Erreur, la méthode privée de la classe de base n'est pas d
      //--- Lecture du membre de la classe de base
      Print(m_member);    // Erreur, le membre privé de la classe de base n'est pas dis
      Print(Member());    // Aucune erreur, la méthode protected est disponible depuis
     }
  };
```

Dans l'exemple ci-dessu, CBaseClass n'a qu'une seule méthode publique - le constructeur. Les constructeurs sont appelés automatiquement lors de la création d'un objet de classe. Le membre privé m_member et la méthode protected Member() ne peuvent donc pas être appelés depuis l'extérieur. Mais dans le cas d'un héritage public, la méthode Member() de la classe de base sera disponible pour les classes dérivées.

Dans le case d'héritage protégé, tous les membres de la classe de base avec un accès public et protected deviennent protected. Cela signifie que si des membres et méthodes publics de la classe de base étaient accessibles depuis l'extérieur, avec l'héritage protected, ils ne sont disponibles que depuis les classes de la classe dérivée et ses propres classes dérivées.

```
//+------------------------------------------------------------------+
//| Classe d'exemple avec plusieurs types d'accès                    |
//+------------------------------------------------------------------+
class CBaseMathClass
  {
private:              //--- Le membre privé n'est pas disponible depuis les classes dér
   double             m_Pi;
public:               //--- Définir et récupérer une valeur pour m_Pi
   void               SetPI(double v){m_Pi=v;return;};
   double             GetPI(){return m_Pi;};
public:               // Le constructeur de classe est disponible pour tous les membres
                      CBaseMathClass() {SetPI(3.14);  PrintFormat("%s",__FUNCTION__);};
  };
//+------------------------------------------------------------------+
//| Classe dérivée dans laquelle m_Pi ne peut pas être modifié       |
//+------------------------------------------------------------------+
class CProtectedChildClass: protected CBaseMathClass // Héritage protégé
  {
private:
   double             m_radius;
public:               //--- Méthodes publiques de la classe dérivée
   void               SetRadius(double r){m_radius=r; return;};
   double             GetCircleLength(){return GetPI()*m_radius;};
  };
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Lors de la création d'une classe dérivée, le constructeur de la classe de base s
   CProtectedChildClass pt;
//--- Spécifie le rayon
```

© 2000-2024, MetaQuotes Ltd.

```
   pt.SetRadius(10);
   PrintFormat("Longueur=%G",pt.GetCircleLength());
//--- Si nous décommentons la ligne ci-dessous, nous obtiendrons une erreur au moment
// pt.SetPI(3);

//--- Déclarons maintenant une variable de la classe de base et essayon de définir la
   CBaseMathClass bc;
   bc.SetPI(10);
//--- Voici le résultat
   PrintFormat("bc.GetPI()=%G",bc.GetPI());
  }
```

L'exemple montre que les méthodes SetPI() et GetPi() de la classe de base CBaseMathClass sont ouvertes et disponibles pour être appelées depuis n'importe où dans le programme. Mais en même temps, pour CProtectedChildClass qui est dérivée, ces méthodes ne peuvent être appelées que depuis les méthodes de la classe CProtectedChildClass ou de ses classes dérivées.

Dans le case d'héritage privé, tous les membres de la classe de base avec un accès public et protected deviennent private, et il devient impossible de les appeler par la suite.

MQL5 ne supporte pas l'héritage multiple.

Voir aussi

Structures et Classes

© 2000-2024, MetaQuotes Ltd.

Polymorphisme

Le polymorphisme est l'opportunité pour différentes classes d'objets, liées entre elles par un héritage, de répondre de différentes façons lors de l'appel à la même fonction. Il aide à créer un mécanisme universel décrivant le comportement non seulement de la classe de base, mais aussi des classes descendantes.

Continuons à développer une classe de base CShape, et définissons une fonction membre GetArea(), conçue pour calculer l'aire d'une forme. Dans toutes les classes descendantes, produites par héritage depuis la classe de base, nous redéfinissons cette fonction selon les règles de calcul de l'aire d'une forme particulière.

Pour un carré (classe CSquare), l'aire est calculée grâce à ses côtés, pour un cercle (class CCircle), l'aire est exprimée grâce à son rayon, etc. Nous pouvons créer un tableau pour stocker des objets de type CShape, dans lesquels à la fois des objets de la classe de base et ceux des classes descendantes peuvent être stockés. Par la suite, nous pourrons appeler la même fonction pour tous les éléments du tableau.

Exemple :

```
//--- Classes de base
class CShape
  {
protected:
   int            m_type;                // Type de la forme
   int            m_xpos;                // X - coordonnées du point de base
   int            m_ypos;                // Y - coordonnées du point de base
public:
   void           CShape(){m_type=0;};   // constructeur, type=0
   int            GetType(){return(m_type);};// retourne le type de la forme
virtual
   double         GetArea(){return (0); }// retourne l'aire de la forme
  };
```

Maintenant, toutes les classes dérivées ont une fonction membre getArea(), qui retourne une valeur zéro. L'implémentation de cette fonction variera dans chaque descendant.

```
//--- La classe dérivée Circle
class CCircle : public CShape            // Après le symbole ':' se trouve la classe d
  {                                      // à partir de laquelle l'héritage est effect
private:
   double         m_radius;              // rayon du cercle

public:
   void           CCircle(){m_type=1;};  // constructeur, type=1
   void           SetRadius(double r){m_radius=r;};
   virtual double GetArea(){return (3.14*m_radius*m_radius);}// aire du cercle
  };
```

Pour la classe Carré, la déclaration est identique :

```
//--- La classe dérivée Carré
```

© 2000-2024, MetaQuotes Ltd.

```
class CSquare : public CShape            // Après le symbole ':' se trouve la classe d
  {                                      // à partir de laquelle l'héritage est effect
private:
   double          m_square_side;        // côté du carré

public:
   void            CSquare(){m_type=2;}; // constructeur, type=1
   void            SetSide(double s){m_square_side=s;};
   virtual double  GetArea(){return (m_square_side*m_square_side);}// aire du carré
  };
```

Pour calculer l'aire du carré et du cercle, nous avons besoin des valeurs correspondantes de m_radius et de m_square_side, nous avons donc ajouté les fonctions SetRadius() et SetSide() dans la déclaration de la classe correspondante.

Il est assumé que des objets de types différents (CCircle et CSquare) dérivés du même type de base CShape, sont utilisés dans notre programme. Le polymorphisme permet de créer un tableau d'objets de la classe de base CShape, mais lors de la déclaration de ce tableau, ces objets ne sont pas encore connus et leurs types sont indéfinis.

La décision sur quel type d'objet sera contenu dans chaque élément du tableau sera prise directement pendant l'exécution du programme. Ceci implique la création dynamique des objets des classes correspondantes, et donc la nécéssité d'utiliser d'utiliser des pointeurs d'objet au lieu des objets directement.

L'opérateur new est utilisé pour créer dynamiquement les objets. Chaque objet doit ensuite être supprimé individuellement et explicitement en utilisant l'opérateur delete. Nous déclarerons donc un tableau de pointeurs de type CShape, et nous créerons un objet du bon type pour chaque élément (**new Class_Name)**, comme montré dans l'exemple du script suivant :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Déclare un tableau de pointeurs d'objets du type de base
   CShape *shapes[5];   // Un tableau de pointeurs d'objets CShape

//--- Remplissage ici du tableau avec les objets dérivés
//--- Déclaration d'un pointeur d'objet de type CCircle
   CCircle *circle=new CCircle();
//--- Définit les propriétés de l'objet du pointeur du cercle
   circle.SetRadius(2.5);
//--- Place la valeur du pointeur dans shapes[0]
   shapes[0]=circle;

//--- Crée un autre objet CCircle et écrit son pointeur dans shapes[1]
   circle=new CCircle();
   shapes[1]=circle;
   circle.SetRadius(5);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Nous "oublions" ici intentionnellement de définir une valeur pour shapes[2]
//circle=new CCircle();
//circle.SetRadius(10);
//shapes[2]=circle;

//--- Affecte NULL pour l'élément qui n'est pas utilisé
   shapes[2]=NULL;

//--- Crée un objet CSquare et écrit son pointeur dans shapes[3]
   CSquare *square=new CSquare();
   square.SetSide(5);
   shapes[3]=square;

//--- Crée un objet CSquare et écrit son pointeur dans shapes[4]
   square=new CSquare();
   square.SetSide(10);
   shapes[4]=square;

//--- Nous avons un tableau de pointeurs, récupérons sa taille
   int total=ArraySize(shapes);
//--- Passe en boucle sur tous les pointeurs du tableau
   for(int i=0; i<5;i++)
     {
      //--- Si le pointeur à l'indice spécifié est valide
      if(CheckPointer(shapes[i])!=POINTER_INVALID)
        {
         //--- Ecrit le type et l'aire de la forme
         PrintFormat("L'objet de type %d à une surface carrée de %G",
               shapes[i].GetType(),
               shapes[i].GetArea());
        }
      //--- Si le pointeur est du type POINTER_INVALID
      else
        {
         //--- Notifie d'une erreur
         PrintFormat("L'objet shapes[%d] n'a pas été initialisé ! Son pointeur est %s'
                     i,EnumToString(CheckPointer(shapes[i])));
        }
     }

//--- Nous devons effacer tous les objets créés dynamiquement
   for(int i=0;i<total;i++)
     {
      //--- Nous ne pouvons supprimer que les objets avec un pointeur de type POINTER_
      if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
        {
         //--- Notifie de la suppression
         PrintFormat("Suppression de shapes[%d]",i);
         //--- Supprime un objet par son pointeur
```

```
            delete shapes[i];
           }
        }
    }
```

Veuillez noter que lors de la suppression d'un objet avec l'opérateur delete, le type de son pointeur doit être vérifié. Seuls les objets avec un pointeur de type POINTER_DYNAMIC peuvent être supprimés avec delete. Pour les pointeurs d'autres types, une erreur sera retournée.

Mais outre la redéfinition des fonctions pendant l'héritage, le polymorphisme inclut également l'implémentation d'une même fonction avec différents ensembles de paramètres dans une classe. Cela signifie que la classe peut avoir plusieurs fonctions avec le même nom mais avec un type différent et/ou un ensemble de paramères différent. Dans ce cas, le polymorphisme est implémenté via la surcharge de fonction.

Voir aussi

Bibliothèque Standard

© 2000-2024, MetaQuotes Ltd.

La surcharge

Dans une classe il est possible de définir au moins deux méthodes qui utilisent le même nom, mais ont de différents nombres de paramètres. Quand cela a lieu, les méthodes s'appellent *surchargées*, et un processus est appellé *la surcharge de méthode*. La surcharge de la méthode – un des moyens, à l'aide duquel se réalise le polymorphisme. La surcharge des méthodes dans les classes est produite selon les mêmes règles que la surcharge des fonctions.

Si pour la fonction appelée il n'y a pas de conformité exacte, le compilateur produit la recherche de la fonction convenante, sur les trois niveaux successivement:

1. la recherche parmi les méthodes de la classe;
2. la recherche parmi les méthodes des classes de base, systématiquement du prédécesseur le plus proche au premier;
3. la recherche parmi les autres fonctions.

S'il n'y a aucune correspondance exacte à tous les niveaux, mais plusieurs fonctions convenables aux différents niveaux ont été trouvées, on utilise la fonction trouvée au plus petit niveau. Dans un niveau, il ne peut pas y avoir plus qu'une fonction convenable.

A MQL5 il n'y a pas de surcharge des opérateurs.

Voir aussi

Surcharge des fonctions

© 2000-2024, MetaQuotes Ltd.

Fonctions virtuelles

Le mot-clé virtual sert de spécificateur pour la fonction, et assure le mécanisme de choix dynamique lors de l'exécution de la fonction-membre qui convient (parmi les fonctions des classes de base et dérivée), les structures ne peuvent pas avoir de fonctions virtuelles. Il peut être utilisé pour changer les déclarations des membres de la fonction seulement.

La fonction virtuelle, comme une fonction ordinaire, doit avoir un corps exécutable. A l'appel, sa sémantique est exactement la même, comme dans les autres fonctions.

La fonction virtuelle peut être remplacée dans la classe dérivée. Le choix de quelle définition de fonction appeler pour une fonction virtuelle, est fait dynamiquement (à l'étape de l'exécution). Un cas typique est quand une classe de base contient une fonction virtuelle et que les classes dérivées ont leurs propres versions de cette fonction.

Le pointeur sur la classe de base peut indiquer l'objet de classe, ou l'objet d'une classe dérivée. Le choix de la fonction-membre appelée sera effectué à l'étape de l'exécution et dépendra du type de l'objet, et non du type du pointeur. S'il n'y a aucun membre d'un type dérivé, une fonction virtuelle de la classe de base est utilisée par défaut.

Les destructeurs sont toujours virtuels, indépendamment du fait qu'ils soient déclarés avec le mot-clé virtual ou non.

Considérons l'utilisation des fonctions virtuelles avec l'exemple du programme MT5_Tetris.mq5. La classe de base CTetrisShape avec une fonction virtuelle Draw (dessiner) est définie dans le fichier inclus MT5_TetrisShape.mqh.

```
//+------------------------------------------------------------------+
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
```

© 2000-2024, MetaQuotes Ltd.

```
   void              Rotate()                    { m_prev_turn=m_turn; if(++m_turn>3) r
   virtual void      Draw()                      { return;                 }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

Ensuite, pour chaque classe dérivée, cette fonction est implémentée conformément aux particularités de la classe-descendante. Par exemple, la première figure CTetrisShape1 a son implémentation de fonction Draw() :

```
class CTetrisShape1 : public CTetrisShape
  {
public:
   //--- dessin de la figure
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      if(m_turn==0 || m_turn==2)
        {
         //--- ligne horizontale
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
           }
        }
      else
        {
         //--- ligne verticale
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+i*SHAPE_SIZE);
           }
        }
     }
  }
```

La figure "carré" est décrite par la classe CTetrisShape6 et elle a sa propre implémentation de méthode Draw() :

```
class CTetrisShape6 : public CTetrisShape
  {
public:
   //--- dessin de la figure
```

© 2000-2024, MetaQuotes Ltd.

```
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      for(i=0; i<2; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
        }
      for(i=2; i<4; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+(i-2)*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+SHAPE_SIZE);
        }
     }
  };
```

En fonction de l'objet, une fonction virtuelle de n'importe quelle classe dérivée est appelée.

```
void CTetrisField::NewShape()
  {
//--- créons une de 7 figures possibles par hasard

   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- dessin
   m_shape.Draw();
//---
  }
```

Modifier 'override'

Le modificateur 'override' signifie que la fonction déclarée doit dériver la méthode de sa classe parente. L'utilisation de ce mot-clé vous permet d'éviter des erreurs de dérivation, par exemple il vous permet d'éviter la modification accidentelle de la signature de la méthode. Supposons que la méthode 'func' est définie dans la classe de base. La méthode accepte une variable de type int comme argument :

© 2000-2024, MetaQuotes Ltd.

```
class CFoo
  {
   void virtual func(int x) const { }
  };
```

Ensuite, la méthode est dérivée dans la classe fille :

```
class CBar : public CFoo
  {
   void func(short x) { }
  };
```

Cependant, le type de l'argument a été changé par erreur de int en short. De fait, ce n'est pas une dérivation de méthode mais une surcharge de méthode. Conformément à l'algorithme de définition des fonctions surchargées, le compilateur peut dans certains cas choisir une méthode définie dans la classe de base au lieu de la méthode dérivée.

Pour éviter ce genre d'erreurs, vous devriez ajouter explicitement le modificateur 'override' à la méthode que vous souhaitez dériver.

```
class CBar : public CFoo
  {
   void func(short x) override { }
  };
```

Si la signature de la métode est changée pendant la dérivation, le compilateur ne pourra pas trouver une méthode avec la même signature dans la classe parente, et il retournera une erreur de compilation :

```
La méthode 'CBar::func' est déclarée avec le spécificateur 'override' mais ne dérive c
```

Modificateur final

Le modificateur 'final' effectue l'inverse — il interdit tout dérivation de la méthode dans les classes filles. Si l'implémentation d'une méthode est suffisante et complète, déclarez cette méthode avec le modificateur 'final' pour être sûr qu'elle ne sera pas modifiée ultérieurement.

```
class CFoo
  {
   void virtual func(int x) final { }
  };

class CBar : public CFoo
  {
   void func(int) { }
  };
```

Si vous essayez de dériver une méthode ayant le modificateur 'final' comme montré dans l'exemple ci-dessus, le compilateur retournera une erreur :

```
La méthode 'CFoo::func' déclarée comme 'final' ne peut pas être dérivée par 'CBar::fun
voir la déclaration de 'CFoo::func'
```

Voir aussi

Bibliothèque standard

© 2000-2024, MetaQuotes Ltd.

Membres Statiques d'une Classe/Structure

Membres Statiques

Les membres d'une classe peuvent être déclarés en utilisant le modificateur de classe de stockage static. Ces membres sont partagés par toutes les instances de cette classe et sont stockés dans un seul endroit. Les membres de données non-statiques sont créés pour chaque variable d'objet de classe.

L'incapacité de déclarer des membres statiques d'une classe aurait mené au besoin de déclarer ces données au niveau global du programme. Cela casserait la relation entre les données et leur classe, et ce n'est pas consistent avec le paradigme de base de la POO - joindre les données et les méthodes pour les gérer dans une classe. Le membre statique permet aux données de classe qui ne sont pas spécifiques à une instance particulière d'exister dans la portée de la classe.

Puiqu'un membre statique de classe ne dépend pas d'une instance particulière, la référence sur lui sera du format suivant :

```
class_name::variable
```

où *class_name* est le nom de la classe, et *variable* est le nom du membre de la classe.

Comme vous le voyez, pour accéder au membre statique d'une classe, l'opérateur de résolution de contexte :: est utilisé. Lorsque vous accédez à un membre statique dans des méthodes de la classe, l'opérateur de contexte est optionnel.

Le membre statique d'une classe doit être initialisé explicitement avec la valeur désirée. Pour cela, il doit être déclaré et initialisé au niveau de la portée globale. La suite d'initialisation des membres statiques correspondra à la suite de leur déclaration dans la portée globale.

Par exemple, nous avons la classe *CParser* utilisée pour parcourir le texte, et nous avons besoin de compter le nombre total de mots et caractères traités. Nous n'avons qu'à déclarer les membres de classe nécessaires comme static et les initialiser au niveau global. Ensuite toutes les instances de la classe utiliseront un compteur commun pour les mots et les caractères.

```
//+------------------------------------------------------------------+
//| Classe "Analyseur de texte"                                      |
//+------------------------------------------------------------------+
class CParser
  {
public:
   static int        s_words;
   static int        s_symbols;
   //--- Constructeur et destructeur
                     CParser(void);
                    ~CParser(void){};
  };
...
//--- Initialisation des membres statiques de la classe Parser au niveau global
int CParser::s_words=0;
int CParser::s_symbols=0;
```

© 2000-2024, MetaQuotes Ltd.

Un membre statique de classe peut être déclaré avec le mot-clé *const*. Ce type de constante statique doit être initialisée au niveau global avec le mot-clé *const* :

```
//+------------------------------------------------------------------+
//| Classe "Stack" pour stocker les données traitées                 |
//+------------------------------------------------------------------+
class CStack
  {
public:
                      CStack(void);
                     ~CStack(void){};
...
private:
   static const int  s_max_length; // Capacité maximum de la pile
  };

//--- Initialisation de la constante statique de la classe CStack
const int CStack::s_max_length=1000;
```

Pointeur this

Le mot-clé this dénote un pointeur déclaré implicitement sur lui-même - vers une instance spécifique de la classe, dans le contexte de laquelle la méthode est exécutée. Il ne peut être utilisé que dans les méthodes non-statiques de la classe. Le pointeur this est un membre implicite non-statique de n'importe quelle classe.

Dans les fonctions statiques, vous ne pouvez accéder qu'aux membres/méthodes statiques d'une classe.

Méthodes Statiques

En MQL5, les fonctions membres de type static peuvent être utilisées. Le modificateur *static* doit précéder le type de retour d'une fonction dans la déclaration dans une classe.

```
class CStack
  {
public:
   //--- Constructeur et destructeur
                      CStack(void){};
                     ~CStack(void){};
   //--- Capacité maximum de la pile
   static int         Capacity();
private:
   int                m_length;     // Le nombre d'élément dans la pile
   static const int  s_max_length; // Capacité maximum de la pile
  };
//+------------------------------------------------------------------+
//| Retourne le nombre maximum d'éléments à stocker sur la pile      |
//+------------------------------------------------------------------+
int CStack::Capacity(void)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   return(s_max_length);
  }
//--- Initialisation de la constante statique de la classe CStack
const int CStack::s_max_length=1000;
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare la variable de type CStack
   CStack stack;
//--- appelle la méthode statique de l'objet
   Print("CStack.s_max_length=",stack.Capacity());
//--- elle peut également être appelée de la façon suivante, comme la méthode est stat
   Print("CStack.s_max_length=",CStack::Capacity());
  }
```

Une méthode avec le modificateur const est appelée constante et ne peut pas modifier les membres implicite de sa classe. La déclaration des fonctions constantes d'une classe et des paramètres constants est appelée contrôle de *const-correction*. Grâce à ce contrôle, vous êtes sûrs que le compilateur s'assurera de la cohérence des valeurs des objets et retournera une erreur pendant la compilation s'il y a quelque chose de faux.

Le modificateur const est placé après la liste des arguments dans la déclaration d'une classe. La définition en dehors d'une classe doit également inclure le modificateur *const* :

```
//+------------------------------------------------------------------+
//| Classe "Rectangle"                                               |
//+------------------------------------------------------------------+
class CRectangle
  {
private:
   double            m_width;      // Largeur
   double            m_height;     // Hauteur
public:
   //--- Constructeurs et destructeur
                     CRectangle(void):m_width(0),m_height(0){};
                     CRectangle(const double w,const double h):m_width(w),m_height(h)
                    ~CRectangle(void){};
   //--- Calcul de la surface
   double            Square(void) const;
   static double     Square(const double w,const double h);// { return(w*h); }
  };
//+------------------------------------------------------------------+
//| Retourne l'aire de l'objet "Rectangle"                           |
//+------------------------------------------------------------------+
double CRectangle::Square(void) const
  {
   return(Square(m_width,m_height));
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Retourne le produit de deux variables                            |
//+------------------------------------------------------------------+
static double CRectangle::Square(const double w,const double h)
  {
   return(w*h);
  }
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Crée un rectangle rect avec les côtés égaux à 5 et 6
   CRectangle rect(5,6);
//--- Trouve l'aire du rectangle en utilisant une méthode constante
   PrintFormat("rect.Square()=%.2f",rect.Square());
//--- Trouve le produit des nombres en utilisant la méthode statique de la classe CRec
   PrintFormat("CRectangle::Square(2.0,1.5)=%f",CRectangle::Square(2.0,1.5));
  }
```

Un argument supplémentaire en faveur de l'utilisation du contrôle de constance est le fait que dans ce cas, le compilateur génèrera une optimisation spéciale, par exemple, placera un objet constant en mémoire en lecture seule.

Une fonction statique ne peut pas être déterminée avec le modificateur const, car ce modificateur assure que les membres de l'instance sont constants lors de l'appel à cette fonction. Mais comme mentionné plus haut, la fonction statique ne peut pas accéder aux membres non-statiques de la classe.

Voir aussi

Variables Statiques, Variables, Références. Modificateur & et Mot-Clé this

© 2000-2024, MetaQuotes Ltd.

Templates de fonction

Les fonctions surchargées sont communément utilisées pour effectuer des opérations similaires sur différents types de données. ArraySize() est un exemple simple de ce genre de fonction en MQL5. Elle retourne la taille de n'importe quel type de tableau. En fait, cette fonction système est surchargée et l'implémentation entière de ce genre de surcharge est masquée pour les développeurs d'applications MQL5 :

```
int  ArraySize(
   void&  array[]      // tableau vérifié
   );
```

Cela signifie que le compilateur du langage MQL5 insère l'implémentation nécessaire pour chaque appel à cette fonction. Par exemple, voici comment cela peut être fait pour les tableaux d'entiers :

```
int  ArraySize(
   int&  array[]      // tableau avec des éléments de type int
   );
```

La fonction ArraySize() peut être affichée de la façon suivante pour le tableau de types MqlRates pour utiliser les cotations au format des données historiques :

```
int  ArraySize(
   MqlRates&  array[] // tableau rempli avec des valeurs de type MqlRates
   );
```

Il est donc très pratique d'utiliser la même fonction pour travailler avec des types différents. Cependant, tout le travail préliminaire doit être effectué - la fonction nécessaire doit être surchargée pour tous les types de données pour lesquels elle sera appelée.

Il y a une solution pratique. Si des opérations similaires doivent être exécutées pour chaque type de données, il est possible d'utiliser des templates de fonctions. Dans ce cas, le programmeur doit n'écrire qu'une seule description pour le template de la fonction. Lors de la description du template de cette façon, nous ne devons spécifier que certains paramètres formels au lieu des type de données définis avec lesquels la fonction doit travailler. Le compilateur génèrera automatiquement les différentes fonction pour gérer chaque type nécessaire sur la base des types des arguments utilisés lors des appels à la fonction.

La définition d'un template de fonction commence avec le mot-clé template suivi de la liste des paramètres formels entre chevrons. Chaque paramètre formel est précédé du mot-clé typename. Les types des paramètres formels sont des types intégrés ou définis par l'utilisateur. Ils sont utilisés :

- pour spécifier les types des arguments de la fonction,
- pour spécifier le type de la valeur de retour de la fonction,
- pour déclarer les variables dans la définition de la fonction

Le nombre de paramètres template ne peut pas être supérieur à 8. Chaque paramètre formel dans la définition du template devrait apparaître dans la liste des paramètres de la fonction au moins une fois. Chaque nom de paramètre formel doit être unique.

© 2000-2024, MetaQuotes Ltd.

Ci-dessous se trouve un exemple de template de fonction pour chercher la plus grande valeur dans un tableau de types numériques (nombres entiers et réels) :

```
template<typename T>
T ArrayMax(T &arr[])
  {
   uint size=ArraySize(arr);
   if(size==0) return(0);

   T max=arr[0];
   for(uint n=1;n<size;n++)
      if(max<arr[n]) max=arr[n];
//---
   return(max);
  }
```

Ce template définit la fonction qui chercher la plus grande valeur dans le tableau passé et retourne cette valeur comme résultat. Gardez à l'esprit que la fonction ArrayMaximum() intégrée en MQL5 ne retourne que l'indice de la plus grande valeur qui peut être utilisée pour trouver la valeur elle-même. Par exemple :

```
//--- crée un tableau
   double array[];
   int size=50;
   ArrayResize(array,size);
//---  remplit avec des valeurs aléatoires
   for(int i=0;i<size;i++)
     {
      array[i]=MathRand();
     }

//--- trouve la position de la plus grande valeur du tableau
   int max_position=ArrayMaximum(array);
//--- maintenant, récupère la plus grande valeur elle-même du tableau
   double max=array[max_position];
//--- affiche la valeur trouvée
   Print("Valeur max = ",max);
```

Nous avons donc effectué deux étapes pour trouver la plus grande valeur dans le tableau. Avec le template de fonction ArrayMax(), nous pouvons trouver le résultat du type désiré en passant juste le tableau du type désiré à cette fonction. Cela signifie qu'au lieu des 2 dernière lignes

```
//--- trouve la position de la plus grande valeur du tableau
   int max_position=ArrayMaximum(array);
//--- maintenant, récupère la plus grande valeur elle-même du tableau
   double max=array[max_position];
```

nous pouvons maintenant n'utiliser qu'une seule ligne, dans laquelle le résultat retourné a le même type que le tableau passé à la fonction :

```
//--- trouve la plus grande valeur
   double max=ArrayMax(array);
```

© 2000-2024, MetaQuotes Ltd.

Dans ce cas, le type du résultat retourné par la fonction ArrayMax() correspondra automatiquement au type du tableau.

Utilisez le mot-clé typename pour récupérer le type de l'argument sous la forme d'une chaîne de caractères pour créer des méthodes générales pour pouvoir travailler avec différents types de données. Considérons un exemple spécifique de la fonction qui retourne un type de données sous la forme d'une chaîne de caractères :

```
#include <Trade\Trade.mqh>
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CTrade trade;
   double d_value=M_PI;
   int i_value=INT_MAX;
   Print("d_value: type=",GetTypeName(d_value), ",   valueur=", d_value);
   Print("i_value: type=",GetTypeName(i_value), ",   valeur=", i_value);
   Print("trade: type=",GetTypeName(trade));
//---
  }
//+------------------------------------------------------------------+
//| Le type est retourné sous forme de texte                         |
//+------------------------------------------------------------------+
template<typename T>
string GetTypeName(const T &t)
  {
//--- retourne le type sous forme de texte
   return(typename(T));
//---
  }
```

Les templates de fonction peuvent également être utilisés pour les méthodes de classe, par exemple :

```
class CFile
  {
   ...
public:
   ...
   template<typename T>
   uint WriteStruct(T &data);
  };

template<typename T>
uint CFile::WriteStruct(T &data)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   ...
   return(FileWriteStruct(m_handle,data));
  }
```

Les templates de fonction ne doivent pas être déclarés avec les mots-clés export, virtual et #import.

Surcharge de template de fonction

La surcharge d'une fonction templatée peut être nécessaire parfois. Par exemple, nous avons une fonctions templatée qui écrit la valeur du deuxième paramètre dans le premier en utilisant une conversion de type. MQL5 ne permet pas la conversion du type string en bool. Nous pouvons le faire nous-mêmes – créons la surcharge d'une fonction templatée. Par exemple :

```
//+------------------------------------------------------------------+
//| Template de fonction                                             |
//+------------------------------------------------------------------+
template<typename T1,typename T2>
string Assign(T1 &var1,T2 var2)
  {
   var1=(T1)var2;
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| Surcharge spéciale pour bool+string                              |
//+------------------------------------------------------------------+
string Assign(bool &var1,string var2)
  {
   var1=(StringCompare(var2,"true",false) || StringToInteger(var2)!=0);
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
   int i;
   bool b;
   Print(Assign(i,"test"));
   Print(Assign(b,"test"));
  }
```

Nous pouvons voir comme résultat de l'exécution du code que la fonction templatée Assign() a été utilisée pour la paire int+string, tandis que la version surchargée a déjà été utilisée pour la paire bool+string pendant le deuxième appel.

```
string Assign<int,string>(int&,string)
string Assign(bool&,string)
```

© 2000-2024, MetaQuotes Ltd.

Voir aussi

Surcharge

© 2000-2024, MetaQuotes Ltd.

Avantages des templates

Les templates de fonctions sont utilisés lorsque vous avez besoin d'effectuer des opérations identiques sur différents types de données, par exemple, chercher l'élément maximum dans un tableau. Le principal avantage d'appliquer des templates est que vous n'avez pas à coder une surcharge séparée pour chaque type. Au lieu de déclarer plusieurs surcharges pour chaque type

```
double ArrayMax(double array[])
  {
   ...
  }
int ArrayMax(int array[])
  {
   ...
  }
uint ArrayMax(uint array[])
  {
   ...
  }
long ArrayMax(long array[])
  {
   ...
  }
datetime ArrayMax(datetime array[])
  {
   ...
  }
```

nous n'avons qu'à écrire une seule fonction templatée

```
template<typename T>
T ArrayMax(T array[])
  {
   if(ArraySize()==0)
      return(0);
   uint max_index=ArrayMaximum(array);
   return(array[max_index]);
  }
```

pour l'utiliser dans votre code :

```
double high[];
datetime time[];
....
double max_high=ArrayMax(high);
datetime lasttime=ArrayMax(time);
```

Ici, le paramètre formel *T* spécifiant le type de la donnée utilisée est remplacé avec le type effectivement utilisé lors de la compilation, c'est à dire que le compilateur génère automatiquement une fonction séparée pour chaque type - double, datetime, etc. MQL5 vous permet aussi de développer des templates de classes utilisant tous les avantages de cette approche.

© 2000-2024, MetaQuotes Ltd.

Templates de classe

Un template de classe est déclaré en utilisant le mot-clé template suivi par des chevrons <> donnant la liste de paramètres formels avec le mot-clé typename. Cette entrée informe la compilateur qu'il traite une classe générique avec le paramètre formel *T* définissant un type réel de variable lors de l'implémentation d'une classe. Par exemple, créons une classe vector pour stocker un tableau d'éléments de type *T* :

```
#define TOSTR(x) #x+" "   // macro affichant le nom d'un objet
//+------------------------------------------------------------------+
//| Classe Vector pour stocker des éléments de type T                |
//+------------------------------------------------------------------+
template <typename T>
class TArray
  {
protected:
   T                 m_array[];
public:
   //--- constructeur créant un tableau de 10 éléments par défaut
   void TArray(void){ArrayResize(m_array,10);}
   //--- constructeur créant un tableau ayant la taille spécifiée
   void TArray(int size){ArrayResize(m_array,size);}
   //--- retourne le type et la quantité de données stockées dans l'objet de type TAr
   string Type(void){return(typename(m_array[0])+":"+(string)ArraySize(m_array));};
  };
```

Appliquons maintenant différentes méthodes pour créer trois objets *TArray* dans le programme pour travailler avec différents types

```
void OnStart()
  {
   TArray<double> double_array;   // le vector a une taille par défaut de 10
   TArray<int> int_array(15);     // le vector a une taille de 15
   TArray<string> *string_array;  // pointeur vers un vecteur de TArray<string>
//--- crée un objet dynamique
   string_array=new TArray<string>(20);
//--- affiche le nom d'un objet, le type de donnée et la taille du vecteur dans le Jou
   PrintFormat("%s (%s)",TOSTR(double_array),double_array.Type());
   PrintFormat("%s (%s)",TOSTR(int_array),int_array.Type());
   PrintFormat("%s (%s)",TOSTR(string_array),string_array.Type());
//--- supprime un objet dynamique avant de terminer le programme
   delete(string_array);
  }
```

Résultats de l'exécution du script :

```
  double_array  (double:10)
  int_array  (int:15)
  string_array  (string:20)
```

© 2000-2024, MetaQuotes Ltd.

Nous avons maintenant 3 vecteurs avec des types de données différents : double, int et string.

Les templates de classe sont pratiques pour développer des conteneurs - des objets conçus pour encapsuler d'autres objets de n'importe quel type. Les objets conteneurs sont des collections contenant déjà des objets d'un certain type. Habituellement, travailler avec les données stockées est intégré instantanément au conteneur.

Par exemple, vous pouvez créer un template de classe qui ne permet pas d'accéder à un élément en dehors du tableau, évitant ainsi l'erreur critique "out of range".

```
//+------------------------------------------------------------------+
//| Classe laissant un accès libre à un élément du tableau           |
//+------------------------------------------------------------------+
template<typename T>
class TSafeArray
  {
protected:
   T                 m_array[];
public:
   //--- constructeur par défaut
   void              TSafeArray(void){}
   //--- constructeur pour créer le tableau avec la taille spécifié
   void              TSafeArray(int size){ArrayResize(m_array,size);}
   //--- taille du tableau
   int               Size(void){return(ArraySize(m_array));}
   //--- change la taille du tableau
   int               Resize(int size,int reserve){return(ArrayResize(m_array,size,rese
   //--- libère le tableau
   void              Erase(void){ZeroMemory(m_array);}
   //--- opérateur pour accéder à un élément du tableau par son indice
   T                 operator[](int index);
   //--- opérateur d'assignement pour récupérer tous les éléments du tableau en une se
   void              operator=(const T  &array[]); // T type array
  };
//+------------------------------------------------------------------+
//| Récupération d'un élément par son indice                         |
//+------------------------------------------------------------------+
template<typename T>
T TSafeArray::operator[](int index)
  {
   static T invalid_value;
//---
   int max=ArraySize(m_array)-1;
   if(index<0 || index>=ArraySize(m_array))
     {
      PrintFormat("L'indice %d de %s n'est pas dans l'intervalle (0-%d) !",index, __FU
      return(invalid_value);
     }
//---
   return(m_array[index]);
```

```
  }
//+------------------------------------------------------------------+
//| Assignement du tableau                                           |
//+------------------------------------------------------------------+
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   int size=ArraySize(array);
   ArrayResize(m_array,size);
//--- Le type T doit supporter l'opérateur de copie
   for(int i=0;i<size;i++)
      m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   int copied,size=15;
   MqlRates rates[];
//--- copy le tableau des cotations
   if((copied=CopyRates(_Symbol,_Period,0,size,rates))!=size)
     {
      PrintFormat("CopyRates(%s,%s,0,%d) a retourné le code d'erreur %d",
      _Symbol,EnumToString(_Period),size,GetLastError());
      return;
     }
//--- crée un conteneur et y insère le tableau de valeurs MqlRates
   TSafeArray<MqlRates> safe_rates;
   safe_rates=rates;
   //--- indice dans le tableau
   int index=3;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
   //--- indice en dehors du tableau
   index=size;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
  }
```

Veuillez noter que la déclaration du template doit également être utilisée lors de l'implémentation des méthodes en dehors de la déclaration de la classe :

```
template<typename T>
T TSafeArray::operator[](int index)
  {
   ...
  }
template<typename T>
void TSafeArray::operator=(const T  &array[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   ...
  }
```

Les templates de classe et de fonction vous permettent de définir plusieurs paramètres formels séparés par une virgule, par exemple, la collection Map pour stocker des paires "clé – valeur" :

```
template<typename Key, template Value>
class TMap
  {
   ...
  }
```

Voir aussi

Templates de fonction, Surcharge

© 2000-2024, MetaQuotes Ltd.

Classes Abstraites et Fonctions Virtuelles Pures

Les classes abstraites sont utilisées pour créer des entités génériques que vous souhaitez utiliser pour créer des classes dérivées plus spécifiques. Une classe abstraite ne peut être utilisée que comme classe de base pour d'autres classes, c'est pourquoi il est impossible de créer un objet du type d'une classe abstraite.

Une classe qui contient au moins une fonction virtuelle pure est abstraite. Les classes dérivées d'une classe abstraite doivent donc implémenter toutes ses fonctions virtuelles pures, sinon elles seront également des classes abstraites.

Une fonction virtuelle est déclarée "pure" en utilisant le mot-clé pure. Considérons l'exemple de la classe CAnimal, qiu n'est créée que pour fournir des fonctions communes - les objets du type CAnimal sont trop généraux pour un usage pratique. CAnimal est donc un bon exemple pour une classe abstraite :

```
class CAnimal
  {
public:
                        CAnimal();      // Constructeur
   virtual void         Sound() = 0;    // Une fonction virtuelle pure
private:
   double               m_legs_count;   // Le nombre de pattes d'un animal
  };
```

Ici Sound() est une fonction virtuelle pure, car elle est déclarée avec le spécificateur de fonction virtuelle PURE (=0).

Les fonctions virtuelles pures ne sont que les fonctions virtuelles déclarées avec le spécificateur PURE : (=NULL) ou (=0). Exemple de déclaration et d'utilisation d'une classe abstraite :

```
class CAnimal
  {
public:
   virtual void         Sound()=NULL;    // méthode PURE, doit être surchargée dans la cl
  };
//--- Dérivé d'une classe abstraite
class CCat : public CAnimal
 {
public:
  virtual void          Sound() { Print("Myau"); } // la fonction PURE est surchargée, (
 };

//--- Exemples de mauvaise utilisation
new CAnimal;          // Erreur de 'CAnimal' - le compilateur retourne l'erreur "cannot
CAnimal some_animal; // Erreur de 'CAnimal' - le compilateur retourne l'erreur "cannot

//--- Exemple d'utilisation correcte
new CCat;  // Aucune erreur - la classe CCat n'est pas abstraite
CCat cat;  // Aucune erreur - la classe CCat n'est pas abstraite
```

© 2000-2024, MetaQuotes Ltd.

Restrictions sur les classes abstraites

Si le constructeur d'une classe abstraite appelle une fonction virtuelle pure (directement ou indirectement), le résultat est indéfini.

```
//+------------------------------------------------------------------+
//| Une classe de base abstraite                                     |
//+------------------------------------------------------------------+
class CAnimal
  {
public:
   //--- Une fonction virtuelle pure
   virtual void      Sound(void)=NULL;
   //--- Fonction
   void              CallSound(void) { Sound(); }
   //--- Constructeur
   CAnimal()
    {
     //--- Un appel explicite à la méthode virtuelle
     Sound();
     //--- Un appel implicite (en utilisant une 3ème fonction)
     CallSound();
     //--- Un constructeur et/ou un destructeur appelle toujours ses propres fonctions
     //--- même si elles sont virtuelles et surchargées par une fonction appelée dans
     //--- Si la fonction appelée est virtuelle pure,
     //--- son appel causera une erreur d'exécution critique : "pure virtual function
    }
  };
```

Cependant, les constructeurs et les destructeurs des classes abstraites peuvent appeler d'autres fonctions membres.

© 2000-2024, MetaQuotes Ltd.

Espace de noms

Un namespace (espace de noms) est une zone spécialement déclarée, dans laquelle différents identifiants sont définis : variables, fonctions, classes, etc. Il est défini grâce au mot clé namespace:

```
namespace nom_de_l'_espace {
  // liste de définitions de fonctions, classes et variables
}
```

L'application de "namespace" permet de diviser l'espace de noms global en sous-espaces. Tous les identifiants dans l'espace de noms sont disponibles les uns pour les autres sans spécification. L'opérateur :: (opération de résolution de contexte) est utilisé pour accéder aux membres de l'espace de noms de l'extérieur.

```
namespace ProjectData
{
class DataManager
  {
public:
   void              LoadData() {}
  };
void Func(DataManager& manager) {}
}
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- utilisation du namespace ProjectData
   ProjectData::DataManager mgr;
   mgr.LoadData();
   ProjectData::Func(mgr);
  }
```

Les espaces de noms sont utilisés pour organiser un code sous forme de groupes logiques et pour éviter les conflits de noms qui peuvent se produire lorsque plusieurs bibliothèques sont utilisées dans un programme. Dans de tels cas, chaque bibliothèque peut être déclarée dans son espace de noms pour accéder explicitement aux fonctions et classes nécessaires de chaque bibliothèque.

Un espace de noms peut être déclaré en plusieurs blocs dans un ou plusieurs fichiers. Le compilateur combine toutes les parties ensemble pendant un prétraitement et l'espace de noms résultant contient tous les membres déclarés dans toutes les parties. Supposons que nous ayons une classe A implémentée dans le fichier include Sample.mqh :

```
//+------------------------------------------------------------------+
//|                                                      Sample.mqh |
//+------------------------------------------------------------------+
class A
  {
public:
                     A() {Print(__FUNCTION__);}
```

© 2000-2024, MetaQuotes Ltd.

```
  };
```

Nous voulons utiliser cette classe dans notre projet, mais nous avons déjà une classe A. Pour pouvoir utiliser les deux classes et éviter les conflits d'identifiants, enveloppez simplement le fichier inclus dans un espace de noms :

```
//--- déclaration de la première classe A
class A
  {
public:
                      A() {Print(__FUNCTION__);}
  };
//--- enveloppe la classe A du fichier Sample.mqh dans l'espace de noms Library pour é
namespace Library
{
#include "Sample.mqh"
}
//--- ajout d'une autre classe à l'espace de noms Library
namespace Library
{
class B
  {
public:
                      B() {Print(__FUNCTION__);}
  };
}
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- utilise la classe A de l'espace de noms global
   A a1;
//--- utilise les classes A et B de l'espace de noms Library
   Library::A a2;
   Library::B b;
  }
//+------------------------------------------------------------------+

/*
Résultats :
   A::A
   Library::A::A
   Library::B::B
*/
```

Les namespaces peuvent être imbriqués. Un espace de noms imbriqué a un accès illimité aux membres de son espace parent, mais les membres de l'espace parent n'ont pas un accès illimité à l'espace de noms imbriqué.

© 2000-2024, MetaQuotes Ltd.

```
namespace General
{
int Func();

namespace Details
{
 int Counter;
 int Refresh()  {return Func(); }
}

int GetBars()   {return(iBars(Symbol(), Period()));};
int Size(int i) {return Details::Counter;}
}
```

Espace de noms global

Si l'identifiant n'est pas explicitement déclaré dans l'espace de noms, il est considéré comme faisant partie implicitement de l'espace de noms global. Pour définir explicitement l'identifiant global, utilisez l'opérateur de résolution de portée sans nom. Cela vous permettra de distinguer cet identifiant de tout autre élément portant le même nom situé dans un espace de noms différent. Par exemple, lors de l'import d'une fonction :

```
#import "lib.dll"
int Func();
#import
//+------------------------------------------------------------------+
//|  Une fonction                                                    |
//+------------------------------------------------------------------+
int Func()
  {
   return(0);
  }
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//+--- appelle la fonction importée
   Print(lib::Func());
//+--- appelle notre fonction
   Print(::Func());
  }
```

Dans ce cas, toutes les fonctions importées de la fonction DLL ont été incluses dans l'espace de noms du même nom. Cela a permis au compilateur de déterminer clairement la fonction à appeler.

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Variables Globales, Variables Locales, Portée de Visibilité et Cycle de Vie des Variables, Création et Suppression des Objets

© 2000-2024, MetaQuotes Ltd.

Les constantes, les énumérations et les structures

Pour simplifier l'écriture de programme et rendre des textes du programme plus convenables pour la perception la langage MQL5 fournit les constantes standards prédéterminées et les énumérations. En outre, pour stocker les informations, on utilise les structures de service.

Les constantes standard sont l'analogue des macrosubstitutions et ont le type int.

Les constantes sont groupées à sa destination:

- Les constantes de graphiques sont utilisées au fonctionnement des graphiques de prix: l'ouverture, la navigation, l'installation des paramètres;
- Les constantes des objetssont destinées au traitement des objets graphiques, qu'on peut créer et afficher sur les graphiques;
- Les constantes des indicateurs servent pour le travail avec les indicateurs standard et d'utilisateur;
- Etat d'environnement – décrivent les propriétés du programme mql5, accordent l'information sur le terminal de client, l'instrument commercial et le compte courant commercial;
- Les constantes commerciales permettent de préciser une information diverse en train du commerce;
- Les constantes nommées sont les constantes du langage MQL5;
- Les structures des données décrivent les formats utilisés du stockage des données;
- Les codes des erreurs et des préventions décrivent les messages du compilateur et le message du serveur commercial sur les requêtes commerciales;
- Les constantes de l'entrée/sortie sont destinées au travail avec les fonctions du fichier et la sortie des messages sur l'écran de l'ordinateur par la fonction MessageBox().

© 2000-2024, MetaQuotes Ltd.

Constantes des Graphiques

Les constantes décrivant les différentes propriétés des graphiques sont divisées suivant les groupes:

- Types d'évènements – évènements se produisant lors de l'utilisation des graphiques;
- Périodes des Graphiques – périodes standard prédéfinies;
- Propriétés du graphique – identifiants utilisés comme paramètres des fonctions du graphique;
- Constantes de positionnement - valeur du paramètre de la fonction ChartNavigate();
- Affichage des graphiques - définition de l'apparence du graphique.

© 2000-2024, MetaQuotes Ltd.

Types des Evènements Graphiques

11 types d'évènements peuvent être traités avec la fonction prédéfinie OnChartEvent(). Pour des évènements personnalisés, 65535 identificateurs sont fournis dans l'intervalle CHARTEVENT_CUSTOM à CHARTEVENT_CUSTOM_LAST inclusive. Pour générer un évènement personnalisé, il faut appeler la fonction EventChartCustom().

ENUM_CHART_EVENT

Identifiant	Description
CHARTEVENT_KEYDOWN	Touches
CHARTEVENT_MOUSE_MOVE	Mouvement et clics de la souris (si CHART_EVENT_MOUSE_MOVE=true est défini pour le graphique)
CHARTEVENT_MOUSE_WHEEL	Pressing or scrolling the mouse wheel (if CHART_EVENT_MOUSE_WHEEL=True for the chart)
CHARTEVENT_OBJECT_CREATE	Objet graphique créé (si CHART_EVENT_OBJECT_CREATE=true est défini pour le graphique)
CHARTEVENT_OBJECT_CHANGE	Changement d'une propriété d'un objet graphique via la boîte de dialogue des propriétés
CHARTEVENT_OBJECT_DELETE	Objet graphique effacé (si CHART_EVENT_OBJECT_DELETE=true est défini pour le graphique)
CHARTEVENT_CLICK	Clic sur un graphique
CHARTEVENT_OBJECT_CLICK	Clic sur un objet graphique
CHARTEVENT_OBJECT_DRAG	Drag and drop d'un objet graphique
CHARTEVENT_OBJECT_ENDEDIT	Fin d'édition du texte dans l'objet graphique Champ d'Edition
CHARTEVENT_CHART_CHANGE	Changement de la taille du graphique ou modification des propriétés du graphique via la boîte de dialogue des propriétés
CHARTEVENT_CUSTOM	Numéro initial d'un évènement dans l'intervalle des évènements personnalisés
CHARTEVENT_CUSTOM_LAST	Le numéro final d'un évènement dans l'intervalle des évènements personnalisés

Pour chaque type d'évènement, les paramètres d'entrée de la fonction OnChartEvent() ont des valeurs définies requises pour le traitement de cet évènement. Les évènements et les valeurs passés via ces paramètres sont listés dans le tableau ci-dessous.

© 2000-2024, MetaQuotes Ltd.

Evènement	Valeur du paramètre id	Valeur du paramètre lparam	Valeur du paramètre dparam	Valeur du paramètre sparam
Evènement suite à un appui sur une touche	CHARTEVENT_KEYDOWN	code de la touche	Nombre de répétitions (le nombre de fois où l'appui sur la touche est réalisé comme étant le résultat de l'appui continu sur la touche)	La valeur de type string du masque de bits décrivant le statut des touches du clavier
Evènements souris (si CHART_EVENT_MOUSE_MOVE=true est défini pour le graphique)	CHARTEVENT_MOUSE_MOVE	la coordonnée X	la coordonnée Y	La valeur de type string du masque de bits décrivant le statut des boutons de la souris
Mouse wheel event (if CHART_EVENT_MOUSE_WHEEL= true for the chart)	CHARTEVENT_MOUSE_WHEEL	Flags of states of keys and mouse buttons, the X and Y coordinates of the mouse pointer. See description in the example below	The Delta value of the mouse wheel scroll	—
Evènement de création d'un objet graphique (si CHART_EVENT_OBJECT_CREATE=true est défini pour le graphique)	CHARTEVENT_OBJECT_CREATE	—	—	Nom de l'objet graphique créé
Evènement de changement de propriété d'un objet via la boîte de dialogue des propriétés	CHARTEVENT_OBJECT_CHANGE	—	—	nom de l'objet graphique modifié
Evènement de suppression d'un objet graphique	CHARTEVENT_OBJECT_DELETE	—	—	Nom de l'objet graphique supprimé

© 2000-2024, MetaQuotes Ltd.

Evènement	Valeur du paramètre id	Valeur du paramètre lparam	Valeur du paramètre dparam	Valeur du paramètre sparam
(si CHART_EVENT_OBJECT_DELETE=true est défini pour le graphique)				
Evènement de clic de la souris sur le graphique	CHARTEVENT_CLICK	la coordonnée X	la coordonnée Y	—
Evènement de clic de la souris sur un objet graphique appartenant au graphique	CHARTEVENT_OBJECT_CLICK	la coordonnée X	la coordonnée Y	Nom de l'objet graphique sur lequel l'évènement s'est produit
Evènement de déplacement d'un objet graphique avec la souris	CHARTEVENT_OBJECT_DRAG	—	—	Nom de l'objet graphique déplacé
Evènement de fin d'édition du texte dans la zone de saisie d'un objet graphique Champ d'Edition	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nom de l'objet graphique Champ d'Edition dans lequel l'édition du texte est terminée
Evènement de changement de la taille du graphique ou modification des propriétés du graphique via la boîte de dialogue des Propriétés	CHARTEVENT_CHART_CHANGE	—	—	—
Identificateur de l'évènement utilisateur sous le numéro N	CHARTEVENT_CUSTOM+N	Valeur définie par la fonction EventChartCustom()	Valeur définie par la fonction EventChartCustom()	Valeur définie par la fonction EventChartCustom()

Exemple :

```
#define KEY_NUMPAD_5        12
```

© 2000-2024, MetaQuotes Ltd.

```
#define KEY_LEFT           37
#define KEY_UP             38
#define KEY_RIGHT          39
#define KEY_DOWN           40
#define KEY_NUMLOCK_DOWN   98
#define KEY_NUMLOCK_LEFT  100
#define KEY_NUMLOCK_5     101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP    104
//+------------------------------------------------------------------------------------
//| Fonction d'initialisation de l'expert
//+------------------------------------------------------------------------------------
int OnInit()
  {
//---
   Print("L'expert avec le nom ",MQL5InfoString(MQL5_PROGRAM_NAME)," est en cours d'ex
//--- active les évènements de création d'objet
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- active les évènements de suppression d'objet
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- la mise à jour forcée des propriétés du graphique assure qu'elles seront prises
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------
//| Fonction ChartEvent
//+------------------------------------------------------------------------------------
void OnChartEvent(const int id,         // Identificateur de l'évènement
                  const long& lparam,   // Paramètre de l'évènement de type long
                  const double& dparam, // Paramètre de l'évènement de type double
                  const string& sparam  // Paramètre de l'évènement de type string
                  )
  {
//--- le bouton gauche de la souris a été appuyé sur le graphique
   if(id==CHARTEVENT_CLICK)
     {
      Print("Les coordonnées du clic de la souris sur le graphique sont : x = ",lparam
     }
//--- le clic de souris a eu lieu sur l'objet graphique
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("Le clic de souris a eu lieu sur l'objet nommé '"+sparam+"'");
     }
//--- la touche a été pressée
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch(lparam)
        {
```

```
          case KEY_NUMLOCK_LEFT:  Print("La touche KEY_NUMLOCK_LEFT a été pressée");
          case KEY_LEFT:          Print("La touche KEY_LEFT a été pressée");
          case KEY_NUMLOCK_UP:    Print("La touche KEY_NUMLOCK_UP a été pressée");
          case KEY_UP:            Print("La touche KEY_UP a été pressée");
          case KEY_NUMLOCK_RIGHT: Print("La touche KEY_NUMLOCK_RIGHT a été pressée");
          case KEY_RIGHT:         Print("la touche KEY_RIGHT a été pressée");
          case KEY_NUMLOCK_DOWN:  Print("La touche KEY_NUMLOCK_DOWN a été pressée");
          case KEY_DOWN:          Print("La touche KEY_DOWN a été pressée");
          case KEY_NUMPAD_5:      Print("La touche KEY_NUMPAD_5 a été pressée");
          case KEY_NUMLOCK_5:     Print("La touche KEY_NUMLOCK_5 a été pressée");
          default:                Print("Une touche non listée a été pressée");
         }
       ChartRedraw();
      }
//--- l'objet a été supprimé
   if(id==CHARTEVENT_OBJECT_DELETE)
      {
       Print("L'objet avec le nom ",sparam," a été supprimé");
      }
//--- l'objet a été créé
   if(id==CHARTEVENT_OBJECT_CREATE)
      {
       Print("L'objet avec le nom ",sparam," a été créé");
      }
//--- l'objet a été déplacé ou les coordonnées de son point d'ancrage ont été changées
   if(id==CHARTEVENT_OBJECT_DRAG)
      {
       Print("Les coordonnées du point d'ancrage de l'objet ",sparam," ont été changées
      }
//--- le texte dans la zone d'édition de l'objet a été changé
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      {
       Print("Le texte dans le champ Zone d'Edition de l'objet ",sparam," a été changé'
      }
  }
```

Pour l'évènement CHARTEVENT_MOUSE_MOVE le paramètre **sparam** de type string contient des informations sur l'état des touches du clavier et des boutons de la souris:

Bit	Description
1	Etat du bouton gauche de la souris
2	Etat du bouton droit de la souris
3	Etat de la touche SHIFT
4	Etat de la touche CTRL
5	Etat du bouton du milieu de la souris

© 2000-2024, MetaQuotes Ltd.

Bit	Description
6	Etat du premier bouton supplémentaire de la souris
7	Etat du deuxième bouton supplémentaire de la souris

Exemple :

```
//+------------------------------------------------------------------------------------
//| Fonction d'initialisation de l'expert
//+------------------------------------------------------------------------------------
void OnInit()
  {
//--- active les messages CHART_EVENT_MOUSE_MOVE
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
// --- désactiver le menu contextuel du graphique (à droite)
   ChartSetInteger(0,CHART_CONTEXT_MENU,0);
// --- désactiver le réticule (par le bouton du milieu)
   ChartSetInteger(0,CHART_CROSSHAIR_TOOL,0);
//--- la mise à jour forcée des propriétés du graphique assure qu'elles seront prises
   ChartRedraw();
  }
//+------------------------------------------------------------------------------------
//| MouseState
//+------------------------------------------------------------------------------------
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // souris gauche
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // souris droite
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // souris milieu
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // souris première bouton X
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // souris deuxième bouton X
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // touche shift
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // touche control
   return(res);
  }
//+------------------------------------------------------------------------------------
//| Fonction ChartEvent
//+------------------------------------------------------------------------------------
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

For the CHARTEVENT_MOUSE_WHEEL event, parameters **lparam** and **dparam** contain information about the states of the **Ctrl** and **Shift** keys, of mouse buttons, cursor coordinates and the mouse wheel

© 2000-2024, MetaQuotes Ltd.

scroll value. For a better understanding, run this Expert Advisor on a chart and scroll the mouse wheel, while pressing different buttons and holding down the keys described in the code.

Example of CHARTEVENT_MOUSE_WHEEL event processing:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
init OnInit()
  {
//--- Enabling mouse wheel scrolling messages
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- La mise à jour forcée des propriétés du graphique assure qu'elles seront prises
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- Consider the state of mouse buttons and wheel for this event
      int flg_keys = (int)(lparam>>32);          // The flag of states of the Ctrl and
      int x_cursor = (int)(short)lparam;         // the X coordinate where the mousse
      int y_cursor = (int)(short)(lparam>>16);   // the Y coordinate where the mousse
      int delta    = (int)dparam;                // the total value of mouse scroll, t
      //--- Processing the flag
      string str_keys="";
      if((flg_keys&0x0001)!=0) str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0) str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0) str_keys+="SHIFT ";
      if((flg_keys&0x0008)!=0) str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0) str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0) str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0) str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1) + "'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
     }
  }
//+------------------------------------------------------------------+ /*
   Example of the output
   CHARTEVENT_MOUSE_WHEEL: Ctrl pressed: X=193, Y=445, delta=-120
   CHARTEVENT_MOUSE_WHEEL: Shift pressed: X=186, Y=446, delta=120
   CHARTEVENT_MOUSE_WHEEL:  X=178, Y=447, delta=-120
```

© 2000-2024, MetaQuotes Ltd.

```
CHARTEVENT_MOUSE_WHEEL:  X=231, Y=449, delta=120
CHARTEVENT_MOUSE_WHEEL: MiddleButton pressed: X=231, Y=449, delta=120
CHARTEVENT_MOUSE_WHEEL: LeftButton pressed: X=279, Y=320, delta=-120
CHARTEVENT_MOUSE_WHEEL: RightButton pressed: X=253, Y=330, delta=120  */
```

Voir aussi

Fonctions de Gestion des Evènements, Travailler avec les évènements

© 2000-2024, MetaQuotes Ltd.

Périodes du Graphique

Toutes les périodes prédéfinies des graphiques ont un identifiant unique. L'identifiant PERIOD_CURRENT signifie la période courante du graphique sur lequel le programme mql5 est exécuté.

ENUM_TIMEFRAMES

Identifiant	Description
PERIOD_CURRENT	Période courante
PERIOD_M1	1 minute
PERIOD_M2	2 minutes
PERIOD_M3	3 minutes
PERIOD_M4	4 minutes
PERIOD_M5	5 minutes
PERIOD_M6	6 minutes
PERIOD_M10	10 minutes
PERIOD_M12	12 minutes
PERIOD_M15	15 minutes
PERIOD_M20	20 minutes
PERIOD_M30	30 minutes
PERIOD_H1	1 heure
PERIOD_H2	2 heures
PERIOD_H3	3 heures
PERIOD_H4	4 heures
PERIOD_H6	6 heures
PERIOD_H8	8 heures
PERIOD_H12	12 heures
PERIOD_D1	1 jour
PERIOD_W1	1 semaine
PERIOD_MN1	1 mois

Exemple :

```
   string chart_name="test_Object_Chart";
   Print("Essayons de créer un objet Graphique avec le nom ",chart_name);
//--- Crée un objet si un object n'existe pas déjà avec ce nom
   if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Définit le symbole
   ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- Définit la coordonnée X du point d'ancrage
   ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- Définit la coordonnée Y du point d'ancrage
   ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- Définit la largeur du graphique
   ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- Définit la hauteur
   ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- Définit la période
   ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- Définit l'échelle (de 0 à 5)
   ObjectSetDouble(0,chart_name,OBJPROP_SCALE,4);
//--- Désactive la sélection à la souris
   ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

Identifiants des timeseries

Les identifiants des timeseries sont utilisés dans les fonctions iHighest() et iLowest() functions. Ils sont égaux à une valeur dans l'énumération

ENUM_SERIESMODE

Identifiant	Description
MODE_OPEN	Prix d'ouverture
MODE_LOW	Prix Low
MODE_HIGH	Prix High
MODE_CLOSE	Prix de clôture (close)
MODE_VOLUME	Volume des ticks
MODE_REAL_VOLUME	Volume réel
MODE_SPREAD	Spread

Voir aussi

PeriodSeconds, Période, Date et Heure, Visibilité des objets

© 2000-2024, MetaQuotes Ltd.

Propriétés du Graphique

Les valeurs de l'énumération ENUM_CHART_PROPERTY sont utilisées comme paramètres des fonctions utilisées pour travailler avec les graphiques. L'abréviation r/o dans la colonne "Type de la Propriété" signifie que cette propriété est en lecture seule et ne peut pas être modifiée. L'abréviation w/o dans la colonne "Type de la Propriété" signifie que cette propriété est en écriture seule et qu'elle ne peut pas être récupérée. Lors de l'accès à certaines propriétés, il est nécessaire de spécifier un paramètre supplémentaire permettant d'indiquer le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale.

Les fonctions définissant les propriétés du graphique sont en fait utilisées pour envoyer les commandes de changement au graphique. Si ces fonctions sont excutées avec succès, la commande est ajoutée dans la queue générale des évènements du graphique. Les changements sont appliqués au graphique lors de la prise en charge des évèvenements du graphique de la queue.

N'attendez donc pas une mise à jour visuelle immédiate du graphique après l'appel à ces fonctions. Le graphique est généralement mis à jour automatiquement par le terminal suivant les changements suivants - l'arrivée d'une nouvelle cotation, le redimensionnement de la fenêtre du graphique, etc. Utiliser la fonction ChartRedraw() pour forcer la mise à jour du graphique.

Pour les fonctions ChartSetInteger() et ChartGetInteger()

ENUM_CHART_PROPERTY_INTEGER

Identifiant	Description	Type de la Propriété
CHART_SHOW	Dessin du graphique des prix. Si false, le dessin de n'importe quel attribut du graphique est désactivé et toutes les échelles des bordures du graphique sont supprimées, incluant les échelles du temps et des prix, la barre de navigation rapide, les étiquettes des évènements du Calendrier, les étiquettes des positions, les infobulles des indicateurs et des barres, les sous-fenêtres des indicateurs, les histogrammes de volume, etc. Désactiver le dessin est la parfaite solution pour créer une interface personnalisée d'un programme utilisant les resources graphiques. Les objets graphiques sont toujours dessinés indépendamment de la valeur de la propriété CHART_SHOW.	bool
CHART_IS_OBJECT	Identification de l'objet "Graphique" (OBJ_CHART) - retourne true pour un object graphique. Retourne false pour un graphique réel	bool r/o
CHART_BRING_TO_TOP	Affiche le graphique par-dessus tous les autres graphiques	bool

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
CHART_CONTEXT_MENU	Active/désactive l'accès au menu contextuel avec le clic droit. Lorsque CHART_CONTEXT_MENU=false, seul le menu contextuel du graphique est désactivé. Le menu contextuel des objets du graphique reste disponible.	bool (true par défaut)
CHART_CROSSHAIR_TOOL	Active/désactive l'accès à l'outil Croix avec le bouton du milieu.	bool (true par défaut)
CHART_MOUSE_SCROLL	Défilement horizontal du graphique avec le bouton gauche de la souris. Le défilement vertical est également possible si la valeur de l'une des propriétés suivantes est à true : CHART_SCALEFIX, CHART_SCALEFIX_11 ou CHART_SCALE_PT_PER_BAR Lorsque CHART_MOUSE_SCROLL=false, le défilement du graphique avec la molette de la souris n'est pas disponible.	bool
CHART_EVENT_MOUSE_WHEEL	Envoie les notifications des évènements de la molette de la souris (CHARTEVENT_MOUSE_WHEEL) à tous les programmes mql5 programs sur graphique	bool (true par défaut)
CHART_EVENT_MOUSE_MOVE	Envoie les notifications des mouvements de la souris et des évènements de clics (CHARTEVENT_MOUSE_MOVE) à tous les programmes mql5 sur un graphique	bool
CHART_EVENT_OBJECT_CREATE	Envoi une notification d'évènement de création d'un nouvel objet (CHARTEVENT_OBJECT_CREATE) à tous les programmes mql5 sur un graphique	bool
CHART_EVENT_OBJECT_DELETE	>Envoie une notification d'évènement de suppression d'un objet (CHARTEVENT_OBJECT_DELETE) à tous les programmes mql5 sur un graphique	bool
CHART_MODE	Type du graphique (chandeliers, barres ou ligne brisée)	enum ENUM_CHART_MODE
CHART_FOREGROUND	Graphique des prix au premier plan	bool
CHART_SHIFT	Décalage du graphique des prix du bord droit	bool
CHART_AUTOSCROLL	Déplacement automatique vers la bordure droite du graphique	bool

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
CHART_KEYBOARD_CONTROL	Permet de gérer le graphique avec le clavier ("Début", "Fin", "Page Précedente", "+", "-", "Flèche vers le haut", etc.). Mettre CHART_KEYBOARD_CONTROL à false désactive le défilement et la mise à l'échelle du graphique tout en laissant intacte la possibilité de reçevoir les évènements d'appui sur les touches dans la fonction OnChartEvent().	bool
CHART_QUICK_NAVIGATION	Permet au graphique d'intercepter l'appui sur les touches Espace et Entrée pour activer la barre de navigation rapide. La barre de navigation rapide apparaît automatiquement dans le bas du graphique après avoir double-cliqué avec la souris ou après avoir appuyé sur Espace/Entrée. Elle vous permet de changer rapidement de symbole, de période et de date de la première barre visible.	bool
CHART_SCALE	Echelle	int de 0 à 5
CHART_SCALEFIX	Mode d'échelle fixe	bool
CHART_SCALEFIX_11	Mode d'échelle 1:1 mode	bool
CHART_SCALE_PT_PER_BAR	Echelle à spécifier en points par barre	bool
CHART_SHOW_TICKER	Affiche un ticker du symbole dans le coin supérieur gauche. Configurer CHART_SHOW_TICKER sur 'false' définit également CHART_SHOW_OHLC sur 'false' et désactive OHLC	bool
CHART_SHOW_OHLC	Affiche les valeurs OHLC dans le coin supérieur gauche. Configurer CHART_SHOW_OHLC sur 'true' définit également CHART_SHOW_TICKER sur 'true' et active le ticker	bool
CHART_SHOW_BID_LINE	Affichage des valeurs Bid sous forme d'une ligne horizontale dans un graphique	bool
CHART_SHOW_ASK_LINE	Affichage des valeurs Ask sous forme d'une ligne horizontale dans un graphique	bool
CHART_SHOW_LAST_LINE	Affichage des valeurs Last sous forme d'une ligne horizontale dans un graphique	bool

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
CHART_SHOW_PERIOD_SEP	Affichage de séparateurs verticaux entre des périodes adjacentes	bool
CHART_SHOW_GRID	Affichage de la grille dans un graphique	bool
CHART_SHOW_VOLUMES	Affichage des volumes dans le graphique	enum ENUM_CHART_VOLUME_MODE
CHART_SHOW_OBJECT_DESCR	Affichage des descriptions des objets (non disponible pour tous les objets)	bool
CHART_VISIBLE_BARS	Le nombre de barres du graphique qui peuvent être affichées	int r/o
CHART_WINDOWS_TOTAL	Le nombre total de fenêtres, incluant les sous-fenêtres d'indicateurs	int r/o
CHART_WINDOW_IS_VISIBLE	Visibilité des sous-fenêtres	bool r/o modificateur - numéro de la sous-fenêtre
CHART_WINDOW_HANDLE	Handle de la fenêtre du graphique (HWND)	int r/o
CHART_WINDOW_YDISTANCE	La distance entre la bordure supérieure de la sous-fenêtre de l'indicateur et la bordure supérieure de la fenêtre du graphique principal, selon l'axe vertical Y, en pixels. Dans le cas d'un évènement souris, le coordonnées du curseur sont passées sous la forme de coordonnées de la fenêtre du graphique principal, tandis que les coordonnées des objets graphiques dans une sous-fenêtre sont relatives au coin supérieur gauche de la sous-fenêtre. La valeur est nécessaire pour convertir les coordonnées absolues du graphique principal en coordonnées locales de la sous-fenêtre pour une utilisation correcte des objets graphiques, dont les coordonnées sont relatives au coin supérieur gauche du cadre de la sous-fenêtre.	int r/o modificateur - numéro de la sous-fenêtre
CHART_FIRST_VISIBLE_BAR	numéro de la première barre visible sur le graphique. L'indexation des barres est la même que pour les timeseries.	int r/o
CHART_WIDTH_IN_BARS	Largeur du graphique en barres	int r/o

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
CHART_WIDTH_IN_PIXELS	Largeur du graphique en pixels	int r/o
CHART_HEIGHT_IN_PIXELS	Hauteur du graphique en pixels	int modificateur - numéro de la sous-fenêtre
CHART_COLOR_BACKGROUND	Couleur d'arrière plan du graphique	color
CHART_COLOR_FOREGROUND	Couleur des axes, de l'échelle et de la ligne OHLC	color
CHART_COLOR_GRID	Couleur de la grille	color
CHART_COLOR_VOLUME	Couleur des volumes et des niveaux d'ouverture des positions	color
CHART_COLOR_CHART_UP	Couleur des barres haussières, des ombres et des contours du corps des bougies haussières	color
CHART_COLOR_CHART_DOWN	Couleur des barres baissières, des ombres et des contours du corps des bougies baissières	color
CHART_COLOR_CHART_LINE	Couleur du graphique en ligne brisée et couleur des bougies japonaises en "Doji"	color
CHART_COLOR_CANDLE_BULL	Couleur du corps d'une bougie haussière	color
CHART_COLOR_CANDLE_BEAR	Couleur du corps d'une bougie baissière	color
CHART_COLOR_BID	Couleur du niveau du prix Bid	color
CHART_COLOR_ASK	Couleur du niveau du prix Ask	color
CHART_COLOR_LAST	Couleur de la ligne du prix de la dernière transaction effectuée (Last)	color
CHART_COLOR_STOP_LEVEL	Couleur des niveaux des ordres stop (Stop Loss et Take Profit)	color
CHART_SHOW_TRADE_LEVELS	Affichage des niveaux de trading sur le graphique (niveaux des positions ouvertes, des Stop Loss, des Take Profit et des ordres en attente)	bool
CHART_DRAG_TRADE_LEVELS	Autorisation de glisser les niveaux de trading sur le graphique avec la souris. Le mode de glissement est activé par défaut (valeur true)	bool

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
CHART_SHOW_DATE_SCALE	Affichage de l'échelle du temps sur le graphique	bool
CHART_SHOW_PRICE_SCALE	Affichage de l'échelle des prix sur le graphique	bool
CHART_SHOW_ONE_CLICK	Affichage du panneau de "Trading En Un Clic"sur le graphique	bool
CHART_IS_MAXIMIZED	La fenêtre du graphique est maximisée	bool r/o
CHART_IS_MINIMIZED	La fenêtre du graphique est minimisée	bool r/o
CHART_IS_DOCKED	La fenêtre du graphique est ancrée. Si définie à false, le graphique peut être déplacé en dehors de la zone du terminal	bool
CHART_FLOAT_LEFT	La coordonnée gauche de la fenêtre non-ancrée du graphique, relativement à l'écran virtuel	int
CHART_FLOAT_TOP	La coordonnée supérieure de la fenêtre non-ancrée du graphique, relativement à l'écran virtuel	int
CHART_FLOAT_RIGHT	La coordonnée droite de la fenêtre non-ancrée du graphique, relativement à l'écran virtuel	int
CHART_FLOAT_BOTTOM	La coordonnée inférieure de la fenêtre non-ancrée du graphique, relativement à l'écran virtuel	int

Pour les fonctions ChartSetDouble() et ChartGetDouble()

ENUM_CHART_PROPERTY_DOUBLE

Identifiant	Description	Type de la Propriété
CHART_SHIFT_SIZE	La taille du décalage de la barre zéro depuis le bord droit en pourcentage	double (de 10% à 50%)
CHART_FIXED_POSITION	La position fixée du graphique depuis le bord gauche en pourcentage. La position fixée du graphique est	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
	marquée par un petit triangle gris sur l'axe horizontal du temps. Il n'est affiché que si le défilement automatique du graphique vers la droite à l'arrivée d'un tick est désactivé (voir la propriété CHART_AUTOSCROLL). La barre d'une position fixée reste à la même place lors du zoom.	
CHART_FIXED_MAX	Maximum fixé du graphique	double
CHART_FIXED_MIN	Minimum fixé du graphique	double
CHART_POINTS_PER_BAR	Echelle en points par barre	double
CHART_PRICE_MIN	Minimum du graphique	double r/o modificateur - numéro de la sous-fenêtre
CHART_PRICE_MAX	Maximum du graphique	double r/o modificateur - numéro de la sous-fenêtre

Pour les fonctions ChartSetString() et ChartGetString()

ENUM_CHART_PROPERTY_STRING

Identifiant	Description	Type de la Propriété
CHART_COMMENT	Texte du commentaire dans un graphique	string
CHART_EXPERT_NAME	Le nom d'un Expert Advisor en cours d'exécution sur le graphique ayant	string r/o

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
	le chart_id spécifié	
CHART_SCRIPT_NAME	Le nom d'un script en cours d'exécution sur le graphique ayant le chart_id spécifié	string r/o

Exemple :

```
int chartMode=ChartGetInteger(0,CHART_MODE);
switch(chartMode)
  {
   case(CHART_BARS):    Print("CHART_BARS");   break;
   case(CHART_CANDLES): Print("CHART_CANDLES");break;
   default:Print("CHART_LINE");
  }
bool shifted=ChartGetInteger(0,CHART_SHIFT);
if(shifted) Print("CHART_SHIFT = true");
else Print("CHART_SHIFT = false");
bool autoscroll=ChartGetInteger(0,CHART_AUTOSCROLL);
if(autoscroll) Print("CHART_AUTOSCROLL = true");
else Print("CHART_AUTOSCROLL = false");
int chartHandle=ChartGetInteger(0,CHART_WINDOW_HANDLE);
Print("CHART_WINDOW_HANDLE = ",chartHandle);
int windows=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
Print("CHART_WINDOWS_TOTAL = ",windows);
if(windows>1)
  {
   for(int i=0;i<windows;i++)
     {
      int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,i);
      double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,i);
      double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,i);
      Print(i+": CHART_HEIGHT_IN_PIXELS = ",height," pixels");
      Print(i+": CHART_PRICE_MIN = ",priceMin);
      Print(i+": CHART_PRICE_MAX = ",priceMax);
     }
  }
```

Voir aussi

Exemples d'Utilisation du Graphique

© 2000-2024, MetaQuotes Ltd.

Constantes de Positionnement

Trois identificateurs de l'énumération ENUM_CHART_POSITION sont les valeurs possibles pour le paramètre *position* de la fonction ChartNavigate().

ENUM_CHART_POSITION

Identifiant	Description
CHART_BEGIN	Début du graphique (les prix les plus anciens)
CHART_CURRENT_POS	Position actuelle
CHART_END	Fin du graphique (les derniers prix)

Exemple:

```
long handle=ChartOpen("EURUSD",PERIOD_H12);
if(handle!=0)
  {
   ChartSetInteger(handle,CHART_AUTOSCROLL,false);
   ChartSetInteger(handle,CHART_SHIFT,true);
   ChartSetInteger(handle,CHART_MODE,CHART_LINE);
   ResetLastError();
   bool res=ChartNavigate(handle,CHART_END,150);
   if(!res) Print("Echec de la fonction Navigate. Erreur = ",GetLastError());
   ChartRedraw();
  }
```

© 2000-2024, MetaQuotes Ltd.

Réprésentation Graphique

Les graphies des prix peuvent être affichés de 3 façons:

- sous forme de barres;
- sous forme de chandeliers japonais (ou bougies);
- sous forme d'une ligne brisée.

Le mode d'affichage du graphique des prix est défini avec la fonction ChartSetInteger(chart_handle,CHART_MODE, chart_mode), où chart_mode est l'une des valeurs de l'énumération ENUM_CHART_MODE.

ENUM_CHART_MODE

Identifiant	Description
CHART_BARS	Affichage sous la forme d'une suite de barres
CHART_CANDLES	Affichage sous la forme de chandeliers japonais
CHART_LINE	Affichage sous la forme d'une ligne brisée reliant les prix de Clôture (Close)

Pour spécifier le mode d'affichage des volumes dans le graphique des prix, la fonctionChartSetInteger(chart_handle, CHART_SHOW_VOLUMES, volume_mode) est utilisée, où volume_mode est l'une des valeurs de l'énumération ENUM_CHART_VOLUME_MODE.

ENUM_CHART_VOLUME_MODE

Identifiant	Description
CHART_VOLUME_HIDE	Les volumes ne sont pas affichés
CHART_VOLUME_TICK	Volumes des ticks
CHART_VOLUME_REAL	Volumes des trades

Exemple:

```
//--- Récupère le handle du graphique actuel
   long handle=ChartID();
   if(handle>0) // Si le handle est correct, nous allons le personnaliser
     {
      //--- Désactive le défilement automatique
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- Active le décalage du bord droit du graphique
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- Affichage sous forme de chandeliers
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- Décalage de 100 barres ddu début de l'historique
      ChartNavigate(handle,CHART_CURRENT_POS,100);
      //--- Définit le mode d'affichage des volumes
```

© 2000-2024, MetaQuotes Ltd.

```
    ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
   }
```

Voir aussi

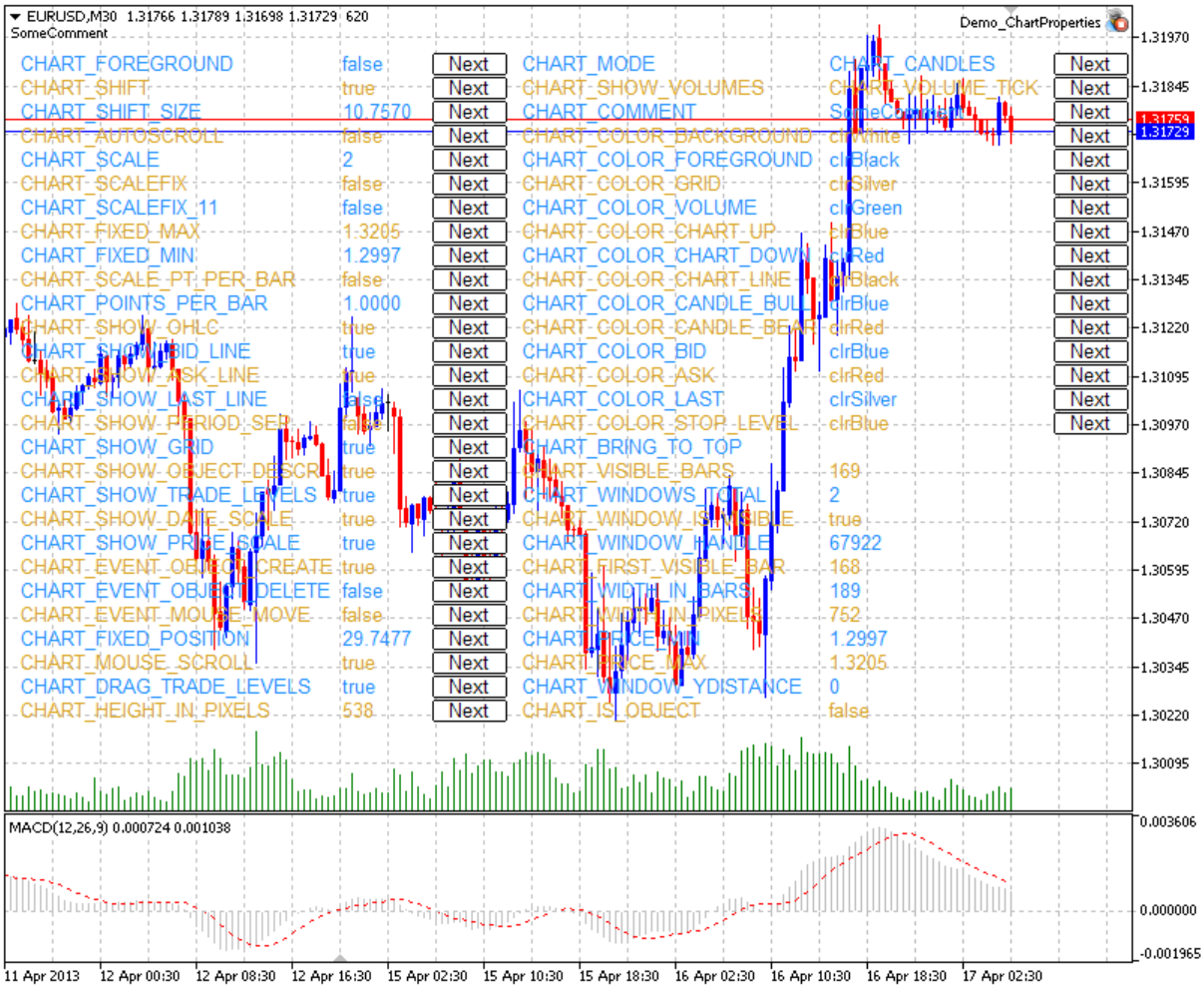
ChartOpen, ChartID

© 2000-2024, MetaQuotes Ltd.

Exemples d'Utilisation du Graphique

Cette section contient des exemples d'utilisation des propriétés du graphique. Une ou deux fonctions complètes sont affichées pour chaque propriété. Ces fonctions permettent de définir/récupérer la valeur de la propriété. Ces fonctions peuvent être utilisées telles quelles dans les applications mql5 personnalisées.

La copie d'écran ci-dessous montre le panneau graphique illustrant comment le changement des propriétés du graphique change son apparence. La nouvelle valeur est définie lors du clic sur le bouton Suivant de la propriété correspondante et le changement est ensuite effectué dans la fenêtre du graphique.



Le code source du panneau est situé en-dessous.

Propriétés du Graphiques et Exemples de Fonctions pour Les utiliser

- **CHART_IS_OBJECT** définit si un objet est un graphique réel ou un objet graphique.

```
//+------------------------------------------------------------------------------------
//| Vérifie si un objet est un graphique. Si c'est un objet graphique,
//| le résultat vaut true. Si c'est un graphique réel, la variable résultat
```

```
//| a la valeur false.
//+------------------------------------------------------------------------------------
bool ChartIsObject(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la propriété du graphique
   if(!ChartGetInteger(chart_ID,CHART_IS_OBJECT,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      //--- retourne false
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
```

- **CHART_BRING_TO_TOP** affiche le graphique par-dessus tous les autres.

```
//+------------------------------------------------------------------------------------
//| Envoie une commande au terminal pour afficher le graphique par-dessus tous les aut
//+------------------------------------------------------------------------------------
bool ChartBringToTop(const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- affiche le graphique par-dessus tous les autres
   if(!ChartSetInteger(chart_ID,CHART_BRING_TO_TOP,0,true))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_MOUSE_SCROLL** est une propriété pour défiler le graphique avec le bouton gauche de la souris.

```
//+------------------------------------------------------------------------------------
//| Vérifie si le défilement du graphique avec le bouton gauche de la souris est activ
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------------------------
bool ChartMouseScrollGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Active/désactive le défilement du graphique avec le bouton gauche de la souris |
//+------------------------------------------------------------------------------------+
bool ChartMouseScrollSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_EVENT_MOUSE_MOVE** est une propriété d'envoi de messages aux application mql5 concernant les évènements de déplacement et de clics de souris (CHARTEVENT_MOUSE_MOVE).

```
//+------------------------------------------------------------------------------------
//| Vérifie si les messages des évènements de déplacement et de clics de souris
//| sont envoyés à toutes les applications MQL5 du graphique
//+------------------------------------------------------------------------------------
bool ChartEventMouseMoveGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive le mode d'envoi des messages concernant les évènements des mouver
//| de clics de souris à toutes les applications MQL5 sur le graphique
//+------------------------------------------------------------------------------------
bool ChartEventMouseMoveSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_EVENT_OBJECT_CREATE** est une propriété d'envoi de messages aux application mql5 concernant les évènements de création d'un objet graphique (CHARTEVENT_OBJECT_CREATE).

```
//+------------------------------------------------------------------------+
//| Vérfie si les messages concernant l'évènement de création d'un objet  |
//| graphique sont envoyés aux applications MQL5 sur le graphique         |
//+------------------------------------------------------------------------+
bool ChartEventObjectCreateGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- affiche le message d'erreur dans le journal des Experts
       Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
       return(false);
      }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive le mode d'envoi des messages concernant les évènements de créatio
//| d'un objet graphique aux applications mql5 sur le graphique
//+------------------------------------------------------------------------------------
bool ChartEventObjectCreateSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_EVENT_OBJECT_DELETE** est une propriété d'envoi de messages aux application mql5 concernant les évènements de suppression d'un objet graphique (CHARTEVENT_OBJECT_DELETE).

```
//+------------------------------------------------------------------------------+
//| Vérifie si les messages concernant l'évènement de suppression d'un           |
//| objet graphique sont envoyés aux applications mql5 sur le graphique          |
//+------------------------------------------------------------------------------+
bool ChartEventObjectDeleteGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive le mode d'envoi des messages concernant les évènements de suppres
//| d'un objet graphique aux applications mql5 sur le graphique
//+------------------------------------------------------------------------------------
bool ChartEventObjectDeleteSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_MODE** - type du graphique (chandeliers, barres ou ligne brisée).

```
//+------------------------------------------------------------------------------------
//| Récupère le type d'affichage du graphique (chandeliers, bougies ou ligne brisée)
//+------------------------------------------------------------------------------------
ENUM_CHART_MODE ChartModeGet(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=WRONG_VALUE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_MODE,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((ENUM_CHART_MODE)result);
  }
//+------------------------------------------------------------------------------------
//| Définit le type d'affichage du graphique (chandeliers, bougies ou ligne brisée)
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
bool ChartModeSet(const long value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_MODE,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_FOREGROUND** est une propriété d'affichage du graphique des prix au premier plan.

```
//+------------------------------------------------------------------------------------
//| Vérifie si le graphique des prix est affiché au premier plan
//+------------------------------------------------------------------------------------
bool ChartForegroundGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive l'affichage du graphique des prix au premier plan
//+------------------------------------------------------------------------------------
bool ChartForegroundSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHIFT** - mode de décalage du graphique des prix du bord droite.

```
//+------------------------------------------------------------------------+
//| Vérifie si le décalage du graphique des prix du bord droit est activé |
//+------------------------------------------------------------------------+
bool ChartShiftGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Active/désactive le décalage du graphique des prix du bord droit             |
//+------------------------------------------------------------------------------+
bool ChartShiftSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

- **CHART_AUTOSCROLL** - le mode de décalage automatique du bord droit du graphique.

```
//+------------------------------------------------------------------------------------
//| Vérifie si le défilement automatique du graphique vers la droite à l'arrivée
//| des nouveaux ticks est activé
//+------------------------------------------------------------------------------------
bool ChartAutoscrollGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive le défilement automatique du graphique vers la droite à l'arrivée
//| à l'arrivée des nouveaux ticks
//+------------------------------------------------------------------------------------
bool ChartAutoscrollSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SCALE** - propriété de l'échelle du graphique.

```
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
//| Récupère l'échelle du graphique (de 0 à 5)
//+------------------------------------------------------------------------------
int ChartScaleGet(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=-1;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SCALE,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((int)result);
  }
//+------------------------------------------------------------------------------
//| Définit l'échelle du graphique (de 0 à 5)
//+------------------------------------------------------------------------------
bool ChartScaleSet(const long value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SCALE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SCALEFIX** - le mode d'échelle fixe du graphique.

```
//+------------------------------------------------------------------------------
//| Vérifie si le mode d'échelle fixe est activé
//+------------------------------------------------------------------------------
bool ChartScaleFixGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX,0,value))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- affiche le message d'erreur dans le journal des Experts
       Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
       return(false);
      }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive le mode d'échelle fixe
//+------------------------------------------------------------------------------------
bool ChartScaleFixSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX,0,value))
      {
       //--- affiche le message d'erreur dans le journal des Experts
       Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
       return(false);
      }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SCALEFIX_11** - mode d'échelle 1:1 du graphique.

```
//+------------------------------------------------------------------------------------
//| Vérifie si l'échelle "1:1" est activée
//+------------------------------------------------------------------------------------
bool ChartScaleFix11Get(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
      {
       //--- affiche le message d'erreur dans le journal des Experts
       Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
       return(false);
      }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive le mode d'échelle "1:1"
//+------------------------------------------------------------------------------------
bool ChartScaleFix11Set(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SCALE_PT_PER_BAR** - le mode de spécification de l'échelle du graphique en points par barre.

```
//+------------------------------------------------------------------------------------
//| Vérifie si le mode d'échelle "points par barre" est activé
//+------------------------------------------------------------------------------------
bool ChartScalePerBarGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive le mode d'échelle "points par barre"
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
bool ChartScalePerBarSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHOW_OHLC** - la propriété d'affichage des valeurs OHLC dans le coin supérieur gauche.

```
//+------------------------------------------------------------------------------------
//| Vérifie si l'affichage des valeurs OHLC dans le coin supérieur gauche du graphique
//+------------------------------------------------------------------------------------
bool ChartShowOHLCGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive l'affichage des valeurs OHLC dans le coin supérieur gauche du gra
//+------------------------------------------------------------------------------------
bool ChartShowOHLCSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHOW_BID_LINE** - la propriété d'affichage de la valeur Bid sous forme d'une ligne horizontale sur le graphique.

```
//+------------------------------------------------------------------------------------
//| Vérifie si l'affichage du prix Bid sous forme d'une ligne horizontale est activé
//+------------------------------------------------------------------------------------
bool ChartShowBidLineGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive l'affichage de la ligne Bid sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowBidLineSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
```

- **CHART_SHOW_ASK_LINE** - la propriété d'affichage de la valeur Ask sous forme d'une ligne horizontale sur le graphique.

```
//+------------------------------------------------------------------------------------
//| Vérifie si l'affichage du prix Ask sous forme d'une ligne horizontale est activé
//+------------------------------------------------------------------------------------
bool ChartShowAskLineGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive l'affichage de la ligne Ask sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowAskLineSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHOW_LAST_LINE** - la propriété d'affichage de la valeur Last sous forme d'une ligne horizontale sur le graphique.

© 2000-2024, MetaQuotes Ltd.

```
//+-----------------------------------------------------------------------------------
//| Vérifie si l'affichage du prix Last sous forme d'une ligne horizontale est activé
//+-----------------------------------------------------------------------------------
bool ChartShowLastLineGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Active/désactive l'affichage d'une ligne au prix de la dernière transaction effect
//+-----------------------------------------------------------------------------------
bool ChartShowLastLineSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHOW_PERIOD_SEP** - la propriété d'affichage de séparateurs verticaux entre les périodes adjacentes.

```
//+-----------------------------------------------------------------------------------+
//| Vérifie si l'affichage de séparateurs verticaux entre les périodes est activé    |
//+-----------------------------------------------------------------------------------+
bool ChartShowPeriodSeparatorGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
```

© 2000-2024, MetaQuotes Ltd.

```
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive l'affichage de séparateurs verticaux entre les périodes adjacente
//+------------------------------------------------------------------------------------
bool ChartShowPeriodSepapatorSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHOW_GRID** - la propriété d'affichage de la grille du graphique.

```
//+------------------------------------------------------------------------------------
//| Vérifie si la grille est affichée sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowGridGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
```

© 2000-2024, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive l'affichage de la grille sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowGridSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHOW_VOLUMES** - la propriété d'affichage des volumes sur le graphique.

```
//+------------------------------------------------------------------------------------
//| Vérifie si les volumes sont affichés sur le graphique.
//| Le flag indique le mode d'affichage des volumes
//+------------------------------------------------------------------------------------
ENUM_CHART_VOLUME_MODE ChartShowVolumesGet(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=WRONG_VALUE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_VOLUMES,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((ENUM_CHART_VOLUME_MODE)result);
  }
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
//| Définit le mode d'affichage des volumes sur le graphique
//+-------------------------------------------------------------------------------------
bool ChartShowVolumesSet(const long value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_VOLUMES,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHOW_OBJECT_DESCR** - la propriété d'affichage des descriptions des objets graphiques.

```
//+-----------------------------------------------------------------------------+
//| Vérifie si les descriptions des objets graphiques s'affichent lorsque |
//| la souris passe au-dessus d'eux                                       |
//+-----------------------------------------------------------------------------+
bool ChartShowObjectDescriptionGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+-------------------------------------------------------------------------------------
//| Active/désactive l'affichage des descriptions des objets graphiques lorsque
//| la souris passe au-dessus d'eux
//+-------------------------------------------------------------------------------------
bool ChartShowObjectDescriptionSet(const bool value,const long chart_ID=0)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_VISIBLE_BARS** définit le nombre de barres disponibles pour l'affichage sur le graphique.

```
//+------------------------------------------------------------------------------------
//| Récupère le nombre de barres qui sont affichées (visibles) sur le graphique
//+------------------------------------------------------------------------------------
int ChartVisibleBars(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=-1;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_VISIBLE_BARS,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_WINDOWS_TOTAL** définit le nombre total de fenêtres incluant les sous-fenêtres des indicateurs.

```
//+------------------------------------------------------------------------------------
//| Récupère le nombre total de fenêtres graphiques incluant les sous-fenêtres des inc
//+------------------------------------------------------------------------------------
int ChartWindowsTotal(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=-1;
//--- réinitialise la valeur d'erreur
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WINDOWS_TOTAL,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_WINDOW_IS_VISIBLE** définit la visibilité des sous-fenêtres.

```
//+------------------------------------------------------------------------------------
//| Vérifie si la fenêtre actuelle du graphique ou la sous-fenêtre est visible
//+------------------------------------------------------------------------------------
bool ChartWindowsIsVisible(bool &result,const long chart_ID=0,const int sub_window=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_IS_VISIBLE,sub_window,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
```

- **CHART_WINDOW_HANDLE** retourne le handle du graphique.

```
//+------------------------------------------------------------------------------------
//| Récupère le handle du graphique
//+------------------------------------------------------------------------------------
int ChartWindowsHandle(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=-1;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_HANDLE,0,result))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- affiche le message d'erreur dans le journal des Experts
       Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      }
//--- retourne la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_WINDOW_YDISTANCE** définit la distance en pixels entre le cadre supérieur de la sous-fenêtre d'indicateur et le cadre supérieur de la fenêtre principale du graphique.

```
//+------------------------------------------------------------------------------------
//| Récupère la distance en pixels entre la bordure supérieure de la sous-fenêtre et
//| la bordure supérieure de la fenêtre principale du graphique
//+------------------------------------------------------------------------------------
int ChartWindowsYDistance(const long chart_ID=0,const int sub_window=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=-1;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_YDISTANCE,sub_window,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_FIRST_VISIBLE_BAR** retourne le numéro de la première barre visible sur le graphique (l'indexation des barres correspond aux time series).

```
//+------------------------------------------------------------------------------------
//| Récupère l'index de la première barre visible sur le graphique.
//| L'indexation est comme pour les timeseries : les dernières barres ont les plus pet
//+------------------------------------------------------------------------------------
int ChartFirstVisibleBar(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=-1;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_FIRST_VISIBLE_BAR,0,result))
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_WIDTH_IN_BARS** retourne la largeur du graphique en barres.

```
//+------------------------------------------------------------------------------------
//| Récupère la largeur du graphique (en barres)
//+------------------------------------------------------------------------------------
int ChartWidthInBars(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=-1;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_BARS,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_WIDTH_IN_PIXELS** retourne la largeur du graphique en pixels.

```
//+------------------------------------------------------------------------------------
//| Récupère la largeur du graphique (en pixels)
//+------------------------------------------------------------------------------------
int ChartWidthInPixels(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=-1;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_PIXELS,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((int)result);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
```

- **CHART_HEIGHT_IN_PIXELS** - propriété de hauteur du graphique en pixels.

```
//+------------------------------------------------------------------------------------
//| Récupère la hauteur du graphique (en pixels)
//+------------------------------------------------------------------------------------
int ChartHeightInPixelsGet(const long chart_ID=0,const int sub_window=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long result=-1;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((int)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la hauteur du graphique (en pixels)
//+------------------------------------------------------------------------------------
bool ChartHeightInPixelsSet(const int value,const long chart_ID=0,const int sub_windo
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_BACKGROUND** - couleur d'arrière plan du graphique.

```
//+------------------------------------------------------------------------------------
//| Récupère la couleur d'arrière plan d'un graphique
//+------------------------------------------------------------------------------------
color ChartBackColorGet(const long chart_ID=0)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur d'arrière plan du graphique
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BACKGROUND,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur d'arrière plan du graphique
//+------------------------------------------------------------------------------------
bool ChartBackColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la couleur d'arrière plan du graphique
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BACKGROUND,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_FOREGROUND** - couleur des axes, de l'échelle et de la ligne OHLC.

```
//+------------------------------------------------------------------------------------
//| Récupère la couleur des axes, de l'échelle et de la ligne OHLC
//+------------------------------------------------------------------------------------
color ChartForeColorGet(const long chart_ID=0)
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur des axes, de l'échelle et de la ligne OHLC
   if(!ChartGetInteger(chart_ID,CHART_COLOR_FOREGROUND,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur des axes, de l'échelle et de la ligne OHLC
//+------------------------------------------------------------------------------------
bool ChartForeColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la couleur des axes, de l'échelle et de la ligne OHLC
   if(!ChartSetInteger(chart_ID,CHART_COLOR_FOREGROUND,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_GRID** - couleur de la grille du graphique.

```
//+------------------------------------------------------------------------------------
//| Récupère la couleur de la grille du graphique
//+------------------------------------------------------------------------------------
color ChartGridColorGet(const long chart_ID=0)
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur de la grille du graphique
   if(!ChartGetInteger(chart_ID,CHART_COLOR_GRID,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur de la grille du graphique
//+------------------------------------------------------------------------------------
bool ChartGridColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit la couleur de la grille du graphique
   if(!ChartSetInteger(chart_ID,CHART_COLOR_GRID,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_VOLUME** - couleur des volumes et des niveaux d'ouverture des positions.

```
//+------------------------------------------------------------------------------------
//| Récupère la couleur des volumes et des niveaux d'entrée sur le marché
//+------------------------------------------------------------------------------------
color ChartVolumeColorGet(const long chart_ID=0)
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur des volumes et des niveaux d'entrée sur le marché
   if(!ChartGetInteger(chart_ID,CHART_COLOR_VOLUME,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur des volumes et des niveaux d'entrée sur le marché
//+------------------------------------------------------------------------------------
bool ChartVolumeColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la couleur des volumes et des niveaux d'entrée sur le marché
   if(!ChartSetInteger(chart_ID,CHART_COLOR_VOLUME,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

- **CHART_COLOR_CHART_UP** - couleur d'une barre haussière, de l'ombre et du contour du corps d'une bougie haussière.

```
//+-----------------------------------------------------------------------------------
//| Récupère la couleur d'une barre haussière, de l'ombre et du contour du corps d'une
//+-----------------------------------------------------------------------------------
color ChartUpColorGet(const long chart_ID=0)
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur d'une barre haussière, de l'ombre et du contour du corps d'u
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_UP,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+-----------------------------------------------------------------------------------
//| Définit la couleur d'une barre haussière, de l'ombre et du contour du corps d'une
//+-----------------------------------------------------------------------------------
bool ChartUpColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la couleur d'une barre haussière, de l'ombre et du contour du corps d'un
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_UP,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_CHART_DOWN** - couleur d'une barre baissière, de l'ombre et du contour du corps d'une bougie baissière.

```
//+-----------------------------------------------------------------------------------
//| Récupère la couleur d'une barre baissière, de l'ombre et du contour du corps d'une
//+-----------------------------------------------------------------------------------
color ChartDownColorGet(const long chart_ID=0)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur d'une barre baissière, de l'ombre et du contour du corps d'u
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_DOWN,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur d'une barre baissière, de l'ombre et du contour du corps d'une
//+------------------------------------------------------------------------------------
bool ChartDownColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la couleur d'une barre baissière, de l'ombre et du contour du corps d'un
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_DOWN,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_CHART_LINE** - couleur de la ligne du graphique et des bougies en forme de Doji.

```
//+------------------------------------------------------------------------------------
//| Récupère la couleur de la ligne du graphique et des bougies en forme de Doji
//+------------------------------------------------------------------------------------
color ChartLineColorGet(const long chart_ID=0)
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur de la ligne du graphique et des bougies en forme de Doji
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_LINE,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur de la ligne du graphique et des bougies en forme de Doji
//+------------------------------------------------------------------------------------
bool ChartLineColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la couleur de la ligne du graphique et des bougies en forme de Doji
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_LINE,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BULL** - couleur du corps des chandeliers haussiers.

```
//+------------------------------------------------------------------------------------
//| Récupère la couleur du corps des chandeliers haussiers
//+------------------------------------------------------------------------------------
color ChartBullColorGet(const long chart_ID=0)
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur du corps des chandeliers haussiers
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur du corps des chandeliers haussiers
//+------------------------------------------------------------------------------------
bool ChartBullColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
```

© 2000-2024, MetaQuotes Ltd.

```
   ResetLastError();
//--- définit la couleur du corps des chandeliers haussiers
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BEAR** - couleur du corps des chandeliers baissiers.

```
//+------------------------------------------------------------------------------------
//| Récupère la couleur du corps des chandeliers baissiers
//+------------------------------------------------------------------------------------
color ChartBearColorGet(const long chart_ID=0)
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur du corps des chandeliers baissiers
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur du corps des chandeliers baissiers
//+------------------------------------------------------------------------------------
bool ChartBearColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la couleur du corps des chandeliers baissiers
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
```

- **CHART_COLOR_BID** - couleur de la ligne du prix Bid.

```
//+------------------------------------------------------------------------------------------
//| Récupère la couleur de la ligne du prix Bid
//+------------------------------------------------------------------------------------------
color ChartBidColorGet(const long chart_ID=0)
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur de la ligne du prix Bid
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BID,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------------
//| Définit la couleur de la ligne du prix Bid
//+------------------------------------------------------------------------------------------
bool ChartBidColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la couleur de la ligne du prix Bid
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BID,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_ASK** - couleur de la ligne du prix Ask.

```
//+------------------------------------------------------------------------------------------
//| Récupère la couleur de la ligne du prix Ask
//+------------------------------------------------------------------------------------------
color ChartAskColorGet(const long chart_ID=0)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur de la ligne du prix Ask
   if(!ChartGetInteger(chart_ID,CHART_COLOR_ASK,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur de la ligne du prix Ask
//+------------------------------------------------------------------------------------
bool ChartAskColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la couleur de la ligne du prix Ask
   if(!ChartSetInteger(chart_ID,CHART_COLOR_ASK,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_LAST** - couleur de la ligne du prix de la dernière transaction effectuée (Last).

```
//+------------------------------------------------------------------------------------
//| Récupère la couleur de la ligne du prix de la dernière transaction effectuée
//+------------------------------------------------------------------------------------
color ChartLastColorGet(const long chart_ID=0)
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur de la ligne du prix de la dernière transaction effectuée (La
   if(!ChartGetInteger(chart_ID,CHART_COLOR_LAST,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur de la ligne du prix de la dernière transaction effectuée
//+------------------------------------------------------------------------------------
bool ChartLastColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la couleur de la ligne du prix de la dernière transaction effectuée (Las
   if(!ChartSetInteger(chart_ID,CHART_COLOR_LAST,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_COLOR_STOP_LEVEL** - couleur des niveaux des ordres stop (Stop Loss et Take Profit).

```
//+------------------------------------------------------------------------------------
//| Récupère la couleur des niveaux des Stop Loss et Take Profit
//+------------------------------------------------------------------------------------
color ChartStopLevelColorGet(const long chart_ID=0)
  {
//--- prépare la variable pour recevoir la couleur
   long result=clrNONE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère la couleur des niveaux des ordres stop (Stop Loss et Take Profit)
   if(!ChartGetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------
//| Définit la couleur des niveaux des Stop Loss et Take Profit
//+------------------------------------------------------------------------------------
bool ChartStopLevelColorSet(const color clr,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit la couleur des niveaux des ordres stop (Stop Loss et Take Profit)
   if(!ChartSetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,clr))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHOW_TRADE_LEVELS** - propriété d'affichage des niveaux de trading sur le graphique (niveaux des positions ouvertes, des Stop Loss, Take Profit et des ordres en attente).

```
//+------------------------------------------------------------------------------------
//| Vérifie les niveaux de trading sont affichés sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive l'affichage des niveaux de trading
//+------------------------------------------------------------------------------------
bool ChartShowTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- exécution réussie
   return(true);
  }
```

- **CHART_DRAG_TRADE_LEVELS** - propriété d'activation de la possiblité de glisser les niveaux de trading sur un graphique avec la souris.

```
//+------------------------------------------------------------------------------------
//| Vérifie si le glissement des niveaux de trading avec la souris est autorisé
//+------------------------------------------------------------------------------------
bool ChartDragTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive le glissement des niveaux de trading sur le graphique en utilisan
//+------------------------------------------------------------------------------------
bool ChartDragTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

- **CHART_SHOW_DATE_SCALE** - propriété d'affichage de l'échelle du temps sur le graphique.

```
//+------------------------------------------------------------------------------------
//| Vérifie si l'échelle du temps est affichée sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowDateScaleGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive l'affichage de l'échelle du temps sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowDateScaleSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHOW_PRICE_SCALE** - propriété d'affichage de l'échelle des prix sur le graphique.

```
//+------------------------------------------------------------------------------------
//| Vérifie si l'échelle des prix est affichée sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowPriceScaleGet(bool &result,const long chart_ID=0)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive l'affichage de l'échelle des prix sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowPriceScaleSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHOW_ONE_CLICK** - propriété d'affichage du panneau de "Trading En Un Clic" sur le graphique.

```
//+------------------------------------------------------------------------------------
//| Vérifie si le panneau de "Trading En Un Clic" est affiché sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowOneClickPanelGet(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- affiche le message d'erreur dans le journal des Experts
       Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
       return(false);
      }
//--- stocke la valeur de la propriété du graphique en mémoire
   result=value;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Active/désactive l'affichage du panneau de "Trading En Un Clic"
//| sur le graphique
//+------------------------------------------------------------------------------------
bool ChartShowOneClickPanelSet(const bool value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
      {
       //--- affiche le message d'erreur dans le journal des Experts
       Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
       return(false);
      }
//--- exécution réussie
   return(true);
  }
```

- **CHART_SHIFT_SIZE** - taille du décalage de la barre zéro depuis le bord droit en pourcentage.

```
//+------------------------------------------------------------------------------+
//| Récupère la taille du décalage de la barre zéro depuis le bord droit         |
//| du graphique en pourcentage (de 10% à 50%)                                   |
//+------------------------------------------------------------------------------+
double ChartShiftSizeGet(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer le résultat
   double result=EMPTY_VALUE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_SHIFT_SIZE,0,result))
      {
       //--- affiche le message d'erreur dans le journal des Experts
       Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      }
//--- retourne la valeur de la propriété du graphique
```

© 2000-2024, MetaQuotes Ltd.

```
   return(result);
  }
//+-----------------------------------------------------------------------------------
//| Récupère la taille du décalage de la barre zéro depuis le bord droit |
//| du graphique en pourcentage (de 10% à 50%). |
//| Pour activer le décalage, la propriété CHART_SHIFT doit être à true. |
//+-----------------------------------------------------------------------------------
bool ChartShiftSizeSet(const double value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetDouble(chart_ID,CHART_SHIFT_SIZE,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_FIXED_POSITION** - position fixée du graphique depuis le bord gauche en pourcentage.

```
//+-----------------------------------------------------------------------------------
//| Récupère la localisation de la position fixée du graphique depuis le bord gauche
//+-----------------------------------------------------------------------------------
double ChartFixedPositionGet(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer le résultat
   double result=EMPTY_VALUE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_FIXED_POSITION,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return(result);
  }
//+-----------------------------------------------------------------------------------
//| Récupère la localisation de la position fixée du graphique depuis le bord gauche
//| Pour voir la localisation de la position fixée du graphique, la valeur de la propr
//| doit être à false.
//+-----------------------------------------------------------------------------------
bool ChartFixedPositionSet(const double value,const long chart_ID=0)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetDouble(chart_ID,CHART_FIXED_POSITION,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_FIXED_MAX** - propriété du maximum fixé du graphique.

```
//+------------------------------------------------------------------------------------
//| Récupère la valeur du maximum fixé du graphique
//+------------------------------------------------------------------------------------
double ChartFixedMaxGet(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer le résultat
   double result=EMPTY_VALUE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MAX,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return(result);
  }
//+------------------------------------------------------------------------------------
//| Définit la valeur du maximum fixé du graphique.
//| Pour changer la valeur de la propriété, la propriété CHART_SCALEFIX
//| doit être à true.
//+------------------------------------------------------------------------------------
bool ChartFixedMaxSet(const double value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MAX,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

```
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_FIXED_MIN** - propriété du minimum fixé du graphique.

```
//+------------------------------------------------------------------------------------
//| Récupère la valeur du minimum fixé du graphique
//+------------------------------------------------------------------------------------
double ChartFixedMinGet(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer le résultat
   double result=EMPTY_VALUE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MIN,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return(result);
  }
//+------------------------------------------------------------------------------------
//| Définit la valeur du minimum fixé du graphique.
//| Pour changer la valeur de la propriété, la propriété CHART_SCALEFIX
//| doit être à true.
//+------------------------------------------------------------------------------------
bool ChartFixedMinSet(const double value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MIN,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

- **CHART_POINTS_PER_BAR** - valeur de l'échelle en points par barre.

```
//+------------------------------------------------------------------------------------
//| Récupère la valeur de l'échelle du graphique en points par barre
//+------------------------------------------------------------------------------------
double ChartPointsPerBarGet(const long chart_ID=0)
  {
//--- prépare la variable pour récupérer le résultat
   double result=EMPTY_VALUE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_POINTS_PER_BAR,0,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return(result);
  }
//+------------------------------------------------------------------------------------
//| Définit la valeur de l'échelle du graphique en points par barre
//| Pour voir le résultat du changement de valeur de cette propriété, la valeur
//| la propriété CHART_SCALE_PT_PER_BAR doit être à true.
//+------------------------------------------------------------------------------------
bool ChartPointsPerBarSet(const double value,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetDouble(chart_ID,CHART_POINTS_PER_BAR,value))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_PRICE_MIN** retourne la valeur du minimum du graphique.

```
//+------------------------------------------------------------------------------------
//| Récupère la valeur du minimum du graphique dans la fenêtre principale ou dans une
//+------------------------------------------------------------------------------------
double ChartPriceMin(const long chart_ID=0,const int sub_window=0)
  {
//--- prépare la variable pour récupérer le résultat
```

© 2000-2024, MetaQuotes Ltd.

```
   double result=EMPTY_VALUE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MIN,sub_window,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return(result);
  }
```

- **CHART_PRICE_MAX** retourne la valeur du maximum du graphique.

```
//+------------------------------------------------------------------------------------
//| Récupère la valeur du maximum du graphique dans la fenêtre principale ou dans une
//+------------------------------------------------------------------------------------
double ChartPriceMax(const long chart_ID=0,const int sub_window=0)
  {
//--- prépare la variable pour récupérer le résultat
   double result=EMPTY_VALUE;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MAX,sub_window,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
     }
//--- retourne la valeur de la propriété du graphique
   return(result);
  }
```

- **CHART_COMMENT** - commentaire sur le graphique.

```
//+------------------------------------------------------------------------------------
//| Retourne le commentaire du coin supérieur gauche du graphique
//+------------------------------------------------------------------------------------
bool ChartCommentGet(string &result,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- reçoit la valeur de la propriété
   if(!ChartGetString(chart_ID,CHART_COMMENT,result))
     {
      //--- affiche le message d'erreur dans le journal des Experts
```

© 2000-2024, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------------
//| Retourne le commentaire du coin supérieur gauche du graphique
//+-----------------------------------------------------------------------------------------
bool ChartCommentSet(const string str,const long chart_ID=0)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit la valeur de la propriété
   if(!ChartSetString(chart_ID,CHART_COMMENT,str))
     {
      //--- affiche le message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

- **CHART_IS_MAXIMIZED** - la fenêtre du graphique est maximisée.

```
//+------------------------------------------------------------------+
//| Détermine si la fenêtre actuelle du graphique est maximisée      |
//+------------------------------------------------------------------+
bool ChartWindowsIsMaximized(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur de l'erreur
   ResetLastError();
//--- récupère la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_IS_MAXIMIZED))
     {
      //--- affiche un message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique dans la variable
   result=value;
//--- exécution réussie
   return(true);
  }
```

- **CHART_IS_MINIMIZED** - la fenêtre du graphique est minimisée.

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Détermine si la fenêtre actuelle du graphique est minimisée      |
//+------------------------------------------------------------------+
bool ChartWindowsIsMinimized(bool &result,const long chart_ID=0)
  {
//--- prépare la variable pour récupérer la valeur de la propriété
   long value;
//--- réinitialise la valeur de l'erreur
   ResetLastError();
//--- récupère la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_IS_MINIMIZED))
     {
      //--- affiche un message d'erreur dans le journal des Experts
      Print(__FUNCTION__+", Code d'Erreur = ",GetLastError());
      return(false);
     }
//--- stocke la valeur de la propriété du graphique dans la variable
   result=value;
//--- exécution réussie
   return(true);
  }
```

Panneau des propriétés du graphique

```
//--- inclut la bibliothèque des éléments de contrôle
#include <ChartObjects\ChartObjectsTxtControls.mqh>
//--- constantes prédéfinies
#define X_PROPERTY_NAME_1    10  // coordonnée x du nom de la propriété dans la premi
#define X_PROPERTY_VALUE_1   225 // coordonnée x de la valeur de la propriété dans la
#define X_PROPERTY_NAME_2    345 // coordonnée x du nom de la propriété dans les deux
#define X_PROPERTY_VALUE_2   550 // coordonnée x de la valeur de la propriété dans les
#define X_BUTTON_1           285 // coordonnée x du bouton dans la première colonne
#define X_BUTTON_2           700 // coordonnée x du bouton dans la deuxième colonne
#define Y_PROPERTY_1         30  // coordonnée y du début des première et deuxième co
#define Y_PROPERTY_2         286 // coordonnée y du début de la troisième colonne
#define Y_DISTANCE           16  // distance axiale y entre les lignes
#define LAST_PROPERTY_NUMBER 111 // numéro de la dernière propriété graphique
//--- paramètres d'entrée
input color InpFirstColor=clrDodgerBlue; // Couleur des lignes impaires
input color InpSecondColor=clrGoldenrod; // Couleur des lignes paires
//--- variables et tableaux
CChartObjectLabel  ExtLabelsName[];  // étiquettes pour afficher les noms des propriét
CChartObjectLabel  ExtLabelsValue[]; // étiquettes pour affichers les valeurs des prop
CChartObjectButton ExtButtons[];     // boutons
int                ExtNumbers[];     // indices des propriétés
string             ExtNames[];       // nom des propriétés
uchar              ExtDataTypes[];   // type de données des propriétés (integer, doub
uint               ExtGroupTypes[];  // tableau pour stocker les données appartenant a
uchar              ExtDrawTypes[];   // tableau pour stocker les données du type d'af
double             ExtMaxValue[];    // valeurs maximum possibles des propriétés lors
double             ExtMinValue[];    // valeurs minimum possibles des propriétés lors
```

© 2000-2024, MetaQuotes Ltd.

```
double              ExtStep[];        // pas de changement des propriétés
int                 ExtCount;         // nombre total de propriétés
color               ExtColors[2];     // tableau de couleurs pour l'affichage des ligne
string              ExtComments[2];   // tableau de commentaires (pour la propriété CHA
//+------------------------------------------------------------------------------------
//| Fonction d'initialisation d'un indicateur personnalisé
//+------------------------------------------------------------------------------------
int OnInit()
  {
//--- affichage d'un commentaire sur le graphique
   Comment("SomeComment");
//--- stocke les couleurs dans le tableau pour pouvoir basculer entre elles par la sui
   ExtColors[0]=InpFirstColor;
   ExtColors[1]=InpSecondColor;
//--- stocke les commentaires dans le tableau pour pouvoir basculer entre eux par la s
   ExtComments[0]="FirstComment";
   ExtComments[1]="SecondComment";
//--- prépare et affiche le panneau de contrôle pour gérer les propriétés du graphique
   if(!PrepareControls())
      return(INIT_FAILED);
//--- exécution réussie
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------
//| Fonction de désinitialisation de l'expert
//+------------------------------------------------------------------------------------
void OnDeinit(const int reason)
  {
//--- efface le commentaire sur le graphique
   Comment("");
  }
//+------------------------------------------------------------------------------------
//| Handler d'un évènement du graphique
//+------------------------------------------------------------------------------------
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- vérifie l'évènement de clic sur l'objet graphique
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      //--- sépare le nom de l'objet avec un séparateur
      string obj_name[];
      StringSplit(sparam,'_',obj_name);
      //--- vérifie si l'objet est un bouton
      if(obj_name[0]=="Button")
        {
         //--- récupère l'indice du bouton
```

```
         int index=(int)StringToInteger(obj_name[1]);
         //--- relâche le bouton
         ExtButtons[index].State(false);
         //--- définit la nouvelle valeur de la propriété suivant son type
         if(ExtDataTypes[index]=='I')
            ChangeIntegerProperty(index);
         if(ExtDataTypes[index]=='D')
            ChangeDoubleProperty(index);
         if(ExtDataTypes[index]=='S')
            ChangeStringProperty(index);
        }
     }
//--- redessine les valeurs de la propriété
   RedrawProperties();
   ChartRedraw();
  }
//+------------------------------------------------------------------------------------
//| Change une propriété de type integer du graphique
//+------------------------------------------------------------------------------------
void ChangeIntegerProperty(const int index)
  {
//--- récupère la valeur courante de la propriété
   long value=ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index]);
//--- définit la valeur suivante de la propriété
   switch(ExtDrawTypes[index])
     {
      case 'C':
         value=GetNextColor((color)value);
         break;
      default:
         value=(long)GetNextValue((double)value,index);
         break;
     }
//--- définit la nouvelle valeur de la propriété
   ChartSetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index],0,value);
  }
//+------------------------------------------------------------------------------------
//| Change une propriété de type double du graphique
//+------------------------------------------------------------------------------------
void ChangeDoubleProperty(const int index)
  {
//--- récupère la valeur courante de la propriété
   double value=ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index]);
//--- définit la valeur suivante de la propriété
   value=GetNextValue(value,index);
//--- définit la nouvelle valeur de la propriété
   ChartSetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index],value);
  }
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
//| Change une propriété de type string du graphique
//+-------------------------------------------------------------------------------------
void ChangeStringProperty(const int index)
  {
//--- variable statique pour basculer dans le tableau ExtComments
   static uint comment_index=1;
//--- change l'indice pour récupérer un autre commentaire
   comment_index=1-comment_index;
//--- définit la nouvelle valeur de la propriété
   ChartSetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[index],ExtComments[comment_
  }
//+-------------------------------------------------------------------------------------
//| Récupère la valeur de la propriété suivante
//+-------------------------------------------------------------------------------------
double GetNextValue(const double value,const int index)
  {
   if(value+ExtStep[index]<=ExtMaxValue[index])
      return(value+ExtStep[index]);
   else
      return(ExtMinValue[index]);
  }
//+-------------------------------------------------------------------------------------
//| Récupère la couleur suivante pour une propriété de type color
//+-------------------------------------------------------------------------------------
color GetNextColor(const color clr)
  {
//--- retourne la couleur suivante
   switch(clr)
     {
      case clrWhite: return(clrRed);
      case clrRed:   return(clrGreen);
      case clrGreen: return(clrBlue);
      case clrBlue:  return(clrBlack);
      default:       return(clrWhite);
     }
  }
//+-------------------------------------------------------------------------------------
//| Redessine les valeurs de la propriété
//+-------------------------------------------------------------------------------------
void RedrawProperties(void)
  {
//--- valeur textuelle de la propriété
   string text;
   long   value;
//--- boucle sur les propriétés
   for(int i=0;i<ExtCount;i++)
     {
      text="";
      switch(ExtDataTypes[i])
```

```
        {
         case 'I':
            //--- récupère la valeur courante de la propriété
            if(!ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[i],0,value))
            break;
            //--- propriété de type integer
            switch(ExtDrawTypes[i])
              {
               //--- propriété de type color
               case 'C':
                  text=(string)((color)value);
                  break;
                  //--- propriété de type boolean
               case 'B':
                  text=(string)((bool)value);
                  break;
                  //--- propriété de type énumération ENUM_CHART_MODE
               case 'M':
                  text=EnumToString((ENUM_CHART_MODE)value);
                  break;
                  //--- propriété de type énumération ENUM_CHART_VOLUME_MODE
               case 'V':
                  text=EnumToString((ENUM_CHART_VOLUME_MODE)value);
                  break;
                  //--- propriété de type int
               default:
                  text=IntegerToString(value);
                  break;
              }
            break;
         case 'D':
            //--- propriété de type double
            text=DoubleToString(ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumber
            break;
         case 'S':
            //--- propriété de type string
            text=ChartGetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[i]);
            break;
        }
      //--- affiche la valeur de la propriété
      ExtLabelsValue[i].Description(text);
     }
  }
//+------------------------------------------------------------------------------------
//| Crée le panneau de gestion des propriétés du graphique
//+------------------------------------------------------------------------------------
bool PrepareControls(void)
  {
//--- alloue la mémoire pour les tableaux avec une réserve
```

```
   MemoryAllocation(LAST_PROPERTY_NUMBER+1);
//--- variables
   int i=0;      // variable de boucle
   int col_1=0; // nombre de propriétés dans la première colonne
   int col_2=0; // nombre de propriétés dans la deuxième colonne
   int col_3=0; // nombre de propriétés dans la troisième colonne
//--- nombre actuel de propriétés - 0
   ExtCount=0;
//--- recherche des propriétés dans la boucle
   while(i<=LAST_PROPERTY_NUMBER)
     {
      //--- stocke le numéro actuel de la propriété
      ExtNumbers[ExtCount]=i;
      //--- augmente la valeur de la variable de boucle
      i++;
      //--- vérifie s'il existe une propriété avec ce numéro
      if(CheckNumber(ExtNumbers[ExtCount],ExtNames[ExtCount],ExtDataTypes[ExtCount],Ex
        {
         //--- crée les éléments de contrôle pour la propriété
         switch(ExtGroupTypes[ExtCount])
           {
            case 1:
               //--- crée les étiquettes et un bouton pour la propriété
               if(!ShowProperty(ExtCount,0,X_PROPERTY_NAME_1,X_PROPERTY_VALUE_1,X_BUTT
               return(false);
               //--- le nombre d'éléments dans la première colonne a augmenté
               col_1++;
               break;
            case 2:
               //--- crée les étiquettes et un bouton pour la propriété
               if(!ShowProperty(ExtCount,1,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,X_BUTT
               return(false);
               //--- le nombre d'éléments dans la deuxième colonne a augmenté
               col_2++;
               break;
            case 3:
               //--- ne crée que les étiquettes pour la propriété
               if(!ShowProperty(ExtCount,2,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,0,Y_PR
               return(false);
               //--- le nombre d'éléments dans la troisième colonne a augmenté
               col_3++;
               break;
           }
         //--- définit les valeurs maximum et minimum de la propriété et le pas
         GetMaxMinStep(ExtNumbers[ExtCount],ExtMaxValue[ExtCount],ExtMinValue[ExtCount
         //--- augmente le nombre de propriétés
         ExtCount++;
        }
     }
```

```
//--- libère la mémoire non utilisée par les tableaux
   MemoryAllocation(ExtCount);
//--- redessine les valeurs de la propriété
   RedrawProperties();
   ChartRedraw();
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Alloue la mémoire pour les tableaux
//+------------------------------------------------------------------------------------
void MemoryAllocation(const int size)
  {
   ArrayResize(ExtLabelsName,size);
   ArrayResize(ExtLabelsValue,size);
   ArrayResize(ExtButtons,size);
   ArrayResize(ExtNumbers,size);
   ArrayResize(ExtNames,size);
   ArrayResize(ExtDataTypes,size);
   ArrayResize(ExtGroupTypes,size);
   ArrayResize(ExtDrawTypes,size);
   ArrayResize(ExtMaxValue,size);
   ArrayResize(ExtMinValue,size);
   ArrayResize(ExtStep,size);
  }
//+------------------------------------------------------------------------------------
//| Vérifie si l'indice de la propriété appartient à l'une des
//| énumérations ENUM_CHART_PROPERTIES
//+------------------------------------------------------------------------------------
bool CheckNumber(const int ind,string &name,uchar &data_type,uint &group_type,uchar &d
  {
//--- vérifie si la propriété est de type integer
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_INTEGER)ind);
   if(_LastError==0)
     {
      data_type='I';                      // propriété de l'énumération ENUM_CHART_PRO
      GetTypes(ind,group_type,draw_type); // définit les paramètres d'affichage de la
      return(true);
     }
//--- vérifie si la propriété est de type double
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_DOUBLE)ind);
   if(_LastError==0)
     {
      data_type='D';                      // propriété de l'énumération ENUM_CHART_PRO
      GetTypes(ind,group_type,draw_type); // définit les paramètres d'affichage de la
      return(true);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- vérifie si la propriété est de type string
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_STRING)ind);
   if(_LastError==0)
     {
      data_type='S';                        // propriété de l'énumération ENUM_CHART_PRO
      GetTypes(ind,group_type,draw_type); // définit les paramètres d'affichage de la
      return(true);
     }
//--- la propriété n'appartient à aucune énumeration
   return(false);
  }
//+------------------------------------------------------------------------------------
//| Définit le groupe dans lequel la propriété devrait être stockée,
//| ainsi que son type d'affichage
//+------------------------------------------------------------------------------------
void GetTypes(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- vérifie si la propriété appartient au troisième groupe
//--- les propriétés du troisième groupe sont affichées dans la deuxième colonne à par
   if(CheckThirdGroup(property_number,group_type,draw_type))
      return;
//--- vérifie si la propriété appartient au deuxième groupe
//--- les propriétés du deuxième groupe sont affichées au début de la deuxième colonne
   if(CheckSecondGroup(property_number,group_type,draw_type))
      return;
//--- si vous vous retrouvez ici, la propriété appartient au premier groupe (première
   CheckFirstGroup(property_number,group_type,draw_type);
  }
//+------------------------------------------------------------------------------------
//| Vérifie si la propriété appartient au troisième groupe et
//| définit son type d'affichage si c'est le cas
//+------------------------------------------------------------------------------------
bool CheckThirdGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- vérifie si la propriété appartient au troisième groupe
   switch(property_number)
     {
      //--- propriétés de type boolean
      case CHART_IS_OBJECT:
      case CHART_WINDOW_IS_VISIBLE:
         draw_type='B';
         break;
         //--- propriétés de type integer
      case CHART_VISIBLE_BARS:
      case CHART_WINDOWS_TOTAL:
      case CHART_WINDOW_HANDLE:
      case CHART_WINDOW_YDISTANCE:
      case CHART_FIRST_VISIBLE_BAR:
```

© 2000-2024, MetaQuotes Ltd.

```
      case CHART_WIDTH_IN_BARS:
      case CHART_WIDTH_IN_PIXELS:
         draw_type='I';
         break;
         //--- propriétés de type double
      case CHART_PRICE_MIN:
      case CHART_PRICE_MAX:
         draw_type='D';
         break;
         //--- en fait, cette propriété est une commande pour afficher le graphique pa
         //--- il n'y a pas besoin d'appliquer ce panneau, puisque la fenêtre sera tou
         //--- par-dessus toutes les autres avant qu'on ne l'utilise
      case CHART_BRING_TO_TOP:
         draw_type=' ';
         break;
         //--- la propriété n'appartient pas au troisième groupe
      default:
         return(false);
     }
//--- la propriété appartient au troisième groupe
   group_type=3;
   return(true);
  }
//+-------------------------------------------------------------------------------------
//| Vérifie si la propriété appartient au deuxième groupe et
//| définit son type d'affichage si c'est le cas
//+-------------------------------------------------------------------------------------
bool CheckSecondGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- vérifie si la propriété appartient au deuxième groupe
   switch(property_number)
     {
      //--- propriété de type ENUM_CHART_MODE
      case CHART_MODE:
         draw_type='M';
         break;
         //--- propriété de type ENUM_CHART_VOLUME_MODE
      case CHART_SHOW_VOLUMES:
         draw_type='V';
         break;
         //--- propriété de type string
      case CHART_COMMENT:
         draw_type='S';
         break;
         //--- propriété de type color
      case CHART_COLOR_BACKGROUND:
      case CHART_COLOR_FOREGROUND:
      case CHART_COLOR_GRID:
      case CHART_COLOR_VOLUME:
```

© 2000-2024, MetaQuotes Ltd.

```
      case CHART_COLOR_CHART_UP:
      case CHART_COLOR_CHART_DOWN:
      case CHART_COLOR_CHART_LINE:
      case CHART_COLOR_CANDLE_BULL:
      case CHART_COLOR_CANDLE_BEAR:
      case CHART_COLOR_BID:
      case CHART_COLOR_ASK:
      case CHART_COLOR_LAST:
      case CHART_COLOR_STOP_LEVEL:
         draw_type='C';
         break;
         //--- la propriété n'appartient pas au deuxième groupe
      default:
         return(false);
     }
//--- la propriété appartient au deuxième groupe
   group_type=2;
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Appelé seulement s'il est déjà connu que la propriété n'appartient pas
//| au deuxième ou au troisième groupe de propriétés
//+------------------------------------------------------------------------------------
void CheckFirstGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- la propriété appartient au premier groupe
   group_type=1;
//--- définit le type d'affichage de la propriété
   switch(property_number)
     {
      //--- propriété de type integer
      case CHART_SCALE:
      case CHART_HEIGHT_IN_PIXELS:
         draw_type='I';
         return;
         //--- propriétés de type double
      case CHART_SHIFT_SIZE:
      case CHART_FIXED_POSITION:
      case CHART_FIXED_MAX:
      case CHART_FIXED_MIN:
      case CHART_POINTS_PER_BAR:
         draw_type='D';
         return;
         //--- il ne reste que les propriétés de type boolean
      default:
         draw_type='B';
         return;
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------------------------
//| Crée l'étiquette et le bouton pour la propriété
//+------------------------------------------------------------------------------------
bool ShowProperty(const int ind,const int type,const int x1,const int x2,
                  const int xb,const int y,const bool btn)
  {
//--- tableau statique pour basculer dans le tableau des couleurs ExtColors
   static uint color_index[3]={1,1,1};
//--- change l'indice pour récupérer une autre couleur
   color_index[type]=1-color_index[type];
//--- affiche les étiquettes et un bouton (si btn=true) pour la propriété
   if(!LabelCreate(ExtLabelsName[ind],"name_"+(string)ind,ExtNames[ind],ExtColors[colo
      return(false);
   if(!LabelCreate(ExtLabelsValue[ind],"value_"+(string)ind,"",ExtColors[color_index[t
      return(false);
   if(btn && !ButtonCreate(ExtButtons[ind],(string)ind,xb,y+1))
      return(false);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Crée l'étiquette
//+------------------------------------------------------------------------------------
bool LabelCreate(CChartObjectLabel &lbl,const string name,const string text,
                 const color clr,const int x,const int y)
  {
   if(!lbl.Create(0,"Label_"+name,0,x,y)) return(false);
   if(!lbl.Description(text))             return(false);
   if(!lbl.FontSize(10))                  return(false);
   if(!lbl.Color(clr))                    return(false);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Crée le bouton
//+------------------------------------------------------------------------------------
bool ButtonCreate(CChartObjectButton &btn,const string name,
                  const int x,const int y)
  {
   if(!btn.Create(0,"Button_"+name,0,x,y,50,15)) return(false);
   if(!btn.Description("Suivant"))                  return(false);
   if(!btn.FontSize(10))                         return(false);
   if(!btn.Color(clrBlack))                      return(false);
   if(!btn.BackColor(clrWhite))                  return(false);
   if(!btn.BorderColor(clrBlack))                return(false);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
//| Définit les valeurs maximum et minimum de la propriété et le pas
//+------------------------------------------------------------------------------------
void GetMaxMinStep(const int property_number,double &max,double &min,double &step)
  {
   double value;
//--- définit les valeurs suivant le type de la propriété
   switch(property_number)
     {
      case CHART_SCALE:
         max=5;
         min=0;
         step=1;
         break;
      case CHART_MODE:
      case CHART_SHOW_VOLUMES:
         max=2;
         min=0;
         step=1;
         break;
      case CHART_SHIFT_SIZE:
         max=50;
         min=10;
         step=2.5;
         break;
      case CHART_FIXED_POSITION:
         max=90;
         min=0;
         step=15;
         break;
      case CHART_POINTS_PER_BAR:
         max=19;
         min=1;
         step=3;
         break;
      case CHART_FIXED_MAX:
         value=ChartGetDouble(0,CHART_FIXED_MAX);
         max=value*1.25;
         min=value;
         step=value/32;
         break;
      case CHART_FIXED_MIN:
         value=ChartGetDouble(0,CHART_FIXED_MIN);
         max=value;
         min=value*0.75;
         step=value/32;
         break;
      case CHART_HEIGHT_IN_PIXELS:
         max=700;
         min=520;
```

© 2000-2024, MetaQuotes Ltd.

```
            step=30;
            break;
            //--- valeurs par défaut
        default:
            max=1;
            min=0;
            step=1;
        }
    }
```

© 2000-2024, MetaQuotes Ltd.

Constantes d'Objets

Il existe 44 objets graphiques pouvant être créés et affichés sur le graphique des prix. Toutes les constantes permettant de travailler avec les objets sont divisées en 9 groupes :

- Types d'objets – identifiants des objets graphiques ;
- Propriétés des Objets – définir et récupérer les propriétés des objets graphiques ;
- Méthodes de Liaison avec les Objets – constantes de positionnement des objets sur le graphique ;
- Coin d'ancrage – définition du coin relativement auquel un objet est positionné sur le graphique ;
- Visibilité des objets – définition des périodes dans lesquelles un objet est visible ;
- Niveaux des Vagues d'Elliott – gradation des vagues ;
- Objets de Gann – constantes de tendance pour l'éventail de Gann et pour la grille de Gann ;
- Couleurs web – constantes des couleurs web prédéfinies ;
- Wingdings – codes des caractères de la police de caractère Wingdings.

© 2000-2024, MetaQuotes Ltd.

Types d'Objets

Lorsqu'un objet graphique est créé avec la fonction ObjectCreate(), il est nécessaire de spécifier le type de l'objet en cours de création, qui peut être l'une des valeurs de l'énumération ENUM_OBJECT. D'autres spécifications des propriétés de l'objet sont possibles avec les fonctions liées aux objets graphiques.

ENUM_OBJECT

Identifiant		Description
OBJ_VLINE		Ligne verticale
OBJ_HLINE		Ligne horizontale
OBJ_TREND		Ligne de tendance
OBJ_TRENDBYANGLE		Ligne de tendance angulaire
OBJ_CYCLES		Lignes de cycle
OBJ_ARROWED_LINE		Ligne fléchée
OBJ_CHANNEL		Canal équidistant
OBJ_STDDEVCHANNEL		Canal de Déviation Standard
OBJ_REGRESSION		Canal de régression linéaire
OBJ_PITCHFORK		Fourchette d'Andrews
OBJ_GANNLINE		Ligne de Gann
OBJ_GANNFAN		Eventail de Gann
OBJ_GANNGRID		Grille de Gann
OBJ_FIBO		Retracements de Fibonacci
OBJ_FIBOTIMES		Périodes de Fibonacci
OBJ_FIBOFAN		Eventail de Fibonacci
OBJ_FIBOARC		Arcs de Fibonacci
OBJ_FIBOCHANNEL		Canal de Fibonacci
OBJ_EXPANSION		Expansion de Fibonacci
OBJ_ELLIOTWAVE5		Vague d'impulsion d'Elliot
OBJ_ELLIOTWAVE3		Vague de Correction d'Elliott
OBJ_RECTANGLE		Rectangle
OBJ_TRIANGLE		Triangle
OBJ_ELLIPSE		Ellipse
OBJ_ARROW_THUMB_UP		Symbole Pouce vers le Haut

© 2000-2024, MetaQuotes Ltd.

Identifiant		Description
OBJ_ARROW_THUMB_DOWN		Symbole Pouce en Bas
OBJ_ARROW_UP		Flèche vers le Haut
OBJ_ARROW_DOWN		Flèche vers le Bas
OBJ_ARROW_STOP		Signe Stop
OBJ_ARROW_CHECK		Signe Coche
OBJ_ARROW_LEFT_PRICE		Etiquette de Prix à Gauche
OBJ_ARROW_RIGHT_PRICE		Etiquette de Prix à Droite
OBJ_ARROW_BUY		Signe Achat
OBJ_ARROW_SELL		Signe Vente
OBJ_ARROW		Flèche
OBJ_TEXT		Texte
OBJ_LABEL		Etiquette
OBJ_BUTTON		Bouton
OBJ_CHART		Graphique
OBJ_BITMAP		Bitmap
OBJ_BITMAP_LABEL		Etiquete Bitmap
OBJ_EDIT		Champ d'Edition
OBJ_EVENT		L'objet "Evènement" correspond à un évènement du calendrier économique
OBJ_RECTANGLE_LABEL		L'objet "Etiquette rectangle" permet de créer créer et de conçevoir une interface graphique personnalisée.

© 2000-2024, MetaQuotes Ltd.

OBJ_VLINE

Ligne verticale.

Note

Lors du dessin d'une ligne verticale, il est possible de définir le mode d'affichage de la ligne pour toutes les fenêtres (propriété OBJPROP_RAY).

Exemple

Le script suivant crée et déplace la ligne verticale sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Ligne Verticale\"."
#property description "Le point d'ancrage de la date est défini en pourcentage de"
#property description "la largeur de la fenêtre du graphique en barres."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="VLine";      // Nom de la ligne
input int             InpDate=25;           // Date de l'évènement, %
input color           InpColor=clrRed;      // Couleur de la ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Style de la ligne
input int             InpWidth=3;           // Largeur de ligne
input bool            InpBack=false;        // Ligne d'arrière-plan
input bool            InpSelection=true;    // Mise en surbrillance pour déplacer
input bool            InpRay=true;          // Prolongement de la ligne vers le bas
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpHidden=true;      // Caché dans la liste des objets
input long            InpZOrder=0;         // Priorité du clic de souris
//+-----------------------------------------------------------------------------------
//| Crée une ligne verticale
//+-----------------------------------------------------------------------------------
bool VLineCreate(const long            chart_ID=0,        // identifiant du graphique
                 const string          name="VLine",      // nom de la ligne
                 const int             sub_window=0,      // indice de sous-fenêtre
                 datetime              time=0,            // line time
                 const color           clr=clrRed,        // couleur de la ligne
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // style de la ligne
                 const int             width=1,           // largeur de la ligne
                 const bool            back=false,        // en arrière plan
                 const bool            selection=true,    // mise en surbrillance pour
                 const bool            ray=true,          // prolongement de la ligne
                 const bool            hidden=true,       // caché dans la liste des o
                 const long            z_order=0)         // priorité pour le clic de
  {
//--- si l'heure de la ligne n'est pas définie, la dessine sur la dernière barre
   if(!time)
      time=TimeCurrent();
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée une ligne verticale
   if(!ObjectCreate(chart_ID,name,OBJ_VLINE,sub_window,time,0))
     {
      Print(__FUNCTION__,
            " : impossible de créer une ligne verticale ! Code d'erreur = ",GetLastEr
      return(false);
     }
//--- définit la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style d'affichage de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de la ligne à la souri
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive (false) le mode d'affichage de la ligne dans les sous
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY,ray);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace la ligne verticale
//+------------------------------------------------------------------------------------
bool VLineMove(const long   chart_ID=0,   // identifiant du graphique
               const string name="VLine", // nom de la ligne
               datetime     time=0)       // heure de la ligne
  {
//--- si l'heure de la ligne n'est pas définie, déplace la ligne sur la dernière barre
   if(!time)
      time=TimeCurrent();
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace la ligne verticale
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la ligne verticale ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime la ligne verticale
//+------------------------------------------------------------------------------------
bool VLineDelete(const long   chart_ID=0,   // identifiant du graphique
                 const string name="VLine") // nom de la ligne
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprme la ligne verticale
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : impossible de supprimer la ligne verticale ! Code d'erreur = ",GetLast
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tableau pour stocker les valeurs de date à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage des ligr
   datetime date[];
//--- allocation mémoire
   ArrayResize(date,bars);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- définit les points pour dessiner la ligne
   int d=InpDate*(bars-1)/100;
//--- crée une ligne verticale
   if(!VLineCreate(0,InpName,0,date[d],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpRay,InpHidden,InpZOrder))
      return;
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant la ligne
//--- compteur de boucle
   int h_steps=bars/2;
//--- déplace la ligne
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- déplace le point
      if(!VLineMove(0,InpName,date[d]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0.03 secondes
      Sleep(30);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime le canal du graphique
   VLineDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_HLINE

Ligne Horizontale.

Exemple

Le script suivant crée et déplace une ligne horizontale sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Ligne Horizontale\"."
#property description "Les coordonnées du point d'ancrage sont en pourcentage de la ta
#property description "de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="HLine";      // Nom de la ligne
input int             InpPrice=25;          // Prix de la ligne, %
input color           InpColor=clrRed;      // Couleur de la ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Style de la ligne
input int             InpWidth=3;           // Largeur de ligne
input bool            InpBack=false;        // Ligne d'arrière-plan
input bool            InpSelection=true;    // Mise en surbrillance pour déplacer
input bool            InpHidden=true;       // Caché dans la liste des objets
input long            InpZOrder=0;          // Priorité du clic de souris
//+------------------------------------------------------------------------------
//| Crée une ligne horizontale
//+------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
bool HLineCreate(const long            chart_ID=0,        // identifiant du graphique
                 const string          name="HLine",      // Nom de la ligne
                 const int             sub_window=0,      // indice de sous-fenêtre
                 double                price=0,           // Prix de la ligne
                 const color           clr=clrRed,        // couleur de la ligne
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // style de la ligne
                 const int             width=1,           // largeur de la ligne
                 const bool            back=false,        // en arrière plan
                 const bool            selection=true,    // mise en surbrillance pour
                 const bool            hidden=true,       // caché dans la liste des o
                 const long            z_order=0)         // priorité pour le clic de
  {
//--- si le prix n'est pas défini, la positionne au niveau du prix Bid actuel
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée une ligne horizontale
   if(!ObjectCreate(chart_ID,name,OBJ_HLINE,sub_window,0,price))
     {
      Print(__FUNCTION__,
            ": impossible de créer une ligne horizontale ! Code d'erreur = ",GetLastEr
      return(false);
     }
//--- définit la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style d'affichage de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de la ligne à la souri
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace la ligne horizontale
//+------------------------------------------------------------------------------------
bool HLineMove(const long   chart_ID=0,   // identifiant du graphique
```

```
                const string name="HLine", // Nom de la ligne
                double       price=0)      // Prix de la ligne
  {
//--- si le prix de la ligne n'est pass défini, la déplace au niveau du prix Bid actue
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace la ligne horizontale
   if(!ObjectMove(chart_ID,name,0,0,price))
     {
      Print(__FUNCTION__,
            ": impossible de déplacer la ligne horizontale ! Code d'erreur = ",GetLast
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime une ligne horizontale
//+------------------------------------------------------------------------------------
bool HLineDelete(const long   chart_ID=0,   // identifiant du graphique
                 const string name="HLine") // Nom de la ligne
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime la ligne horizontale
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": impossible de supprimer la ligne horizontale ! Code d'erreur = ",GetLas
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpPrice<0 || InpPrice>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- taille du tableau des prix
   int accuracy=1000;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- tableau pour stocker les valeurs de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage des lig
   double price[];
//--- allocation mémoire
   ArrayResize(price,accuracy);
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner la ligne
   int p=InpPrice*(accuracy-1)/100;
//--- crée une ligne horizontale
   if(!HLineCreate(0,InpName,0,price[p],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant la ligne
//--- compteur de boucle
   int v_steps=accuracy/2;
//--- déplace la ligne
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p<accuracy-1)
         p+=1;
      //--- déplace le point
      if(!HLineMove(0,InpName,price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime du graphique
   HLineDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
```

© 2000-2024, MetaQuotes Ltd.

```
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_TREND

Ligne de tendance.

Note

Pour la "Ligne de Tendance", il est possible de spécifier le mode de prolongement de son affichage vers la droite et/ou vers la gauche (propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT respectivement).

Exemple

Le script suivant crée et déplace une ligne de tendance sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Ligne de Tendance\"."
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="Trend";      // Nom de la ligne
input int             InpDate1=35;          // Date du premier point, %
input int             InpPrice1=60;         // Prix du premier point, %
input int             InpDate2=65;          // Date du second point, %
input int             InpPrice2=40;         // Prix du second point, %
input color           InpColor=clrRed;      // Couleur de la ligne
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Style de la ligne
input int             InpWidth=2;          // Largeur de la ligne
input bool            InpBack=false;       // Ligne d'arrière-plan
input bool            InpSelection=true;   // Mise en surbrillance pour déplacer
input bool            InpRayLeft=false;    // Prolongement de la ligne vers la gauche
input bool            InpRayRight=false;   // Prolongement de la ligne vers la droite
input bool            InpHidden=true;      // Caché dans la liste des objets
input long            InpZOrder=0;         // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée une ligne de tendance aux coordonnées données
//+------------------------------------------------------------------------------------
bool TrendCreate(const long            chart_ID=0,        // identifiant du graphique
                 const string          name="TrendLine",  // nom de la ligne
                 const int             sub_window=0,      // indice de sous-fenêtre
                 datetime              time1=0,           // heure du premier point
                 double                price1=0,          // prix du premier point
                 datetime              time2=0,           // heure du deuxième point
                 double                price2=0,          // prix du deuxième point
                 const color           clr=clrRed,        // couleur de la ligne
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // style de la ligne
                 const int             width=1,           // largeur de la ligne
                 const bool            back=false,        // en arrière plan
                 const bool            selection=true,    // mise en surbrillance pour
                 const bool            ray_left=false,    // Prolongement de la ligne
                 const bool            ray_right=false,   // Prolongement de la ligne
                 const bool            hidden=true,       // caché dans la liste des o
                 const long            z_order=0)         // priorité pour le clic de
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeTrendEmptyPoints(time1,price1,time2,price2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée une ligne de tendance par les coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_TREND,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            " : impossible de créer une ligne de tendance ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- définit la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style d'affichage de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de la ligne à la souri
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
```

```
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage de la l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage de la l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage de la ligne de tendance
//+------------------------------------------------------------------------------------
bool TrendPointChange(const long   chart_ID=0,       // identifiant du graphique
                      const string name="TrendLine", // nom de la ligne
                      const int    point_index=0,    // indice du point d'ancrage
                      datetime     time=0,           // coordonnées du point d'ancrage
                      double       price=0)          // coordonnées du point d'ancrage
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage de la ligne de tendance
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| La fonction supprime la ligne de tendance du graphique
//+------------------------------------------------------------------------------------
bool TrendDelete(const long   chart_ID=0,       // identifiant du graphique
                 const string name="TrendLine") // nom de la ligne
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
```

```
//--- supprime une ligne de tendance
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : impossible de supprimer la ligne de tendance ! Code d'erreur = ",GetLa
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de la ligne de tendance et définit la val
//| par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- si l'heure du premier point n'est pas définie, elle sera définie sur la barre co
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du second point n'est pas réglée, elle se trouve à 9 barres de la sec
   if(!time2)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- Positionne le deuxième point à 9 barres à gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas défini, il est égal à celui du premier po
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
```

© 2000-2024, MetaQuotes Ltd.

```
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage des ligr
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner la ligne
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crée une ligne de tendance
   if(!TrendCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,InpStyle
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplacez les points d'ancrage de la ligne
//--- compteur de boucle
   int v_steps=accuracy/5;
//--- déplace le premier point d'ancrage verticalement
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1>1)
         p1-=1;
      //--- déplace le point
```

© 2000-2024, MetaQuotes Ltd.

```
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- déplacez le second point d'ancrage à la verticale
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p2<accuracy-1)
         p2+=1;
      //--- déplace le point
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- compteur de boucle
   int h_steps=bars/2;
//--- déplacez les deux points d'ancrage horizontalement en même temps
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise les valeurs suivantes
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- décale les points
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0.03 secondes
      Sleep(30);
     }
//--- délai de 1 seconde
```

© 2000-2024, MetaQuotes Ltd.

```
   Sleep(1000);
//--- supprime une ligne de tendance
   TrendDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_TRENDBYANGLE

Ligne de Tendance Angulaire.

Note

Pour la "Ligne de Tendance Angulaire", il est possible de spécifier le mode de prolongement de son affichage vers la droite et/ou vers la gauche (propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT respectivement).

Pour déterminer la pente de la ligne, il est possible d'utiliser un angle ou les coordonnées du deuxième point d'ancrage.

Exemple

Le script suivant crée et déplace une ligne de tendance sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Ligne de Tendance Angulai
#property description "Les coordonnées du point d'ancrage sont en pourcentage de la ta
#property description "de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="Trend";      // Nom de la ligne
input int             InpDate1=50;          // Date du premier point, %
input int             InpPrice1=75;         // Prix du premier point, %
input int             InpAngle=0;           // Angle de pente de la ligne
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;      // Couleur de la ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Style de la ligne
input int             InpWidth=2;           // Largeur de la ligne
input bool            InpBack=false;        // Ligne d'arrière-plan
input bool            InpSelection=true;    // Mise en surbrillance pour déplacer
input bool            InpRayLeft=false;     // Prolongement de la ligne vers la gauche
input bool            InpRayRight=true;     // Prolongement de la ligne vers la droite
input bool            InpHidden=true;       // Caché dans la liste des objets
input long            InpZOrder=0;          // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée une ligne de tendance angulaire
//+------------------------------------------------------------------------------------
bool TrendByAngleCreate(const long            chart_ID=0,        // identifiant du gra
                        const string          name="TrendLine",  // nom de la ligne
                        const int             sub_window=0,      // indice de sous-fen
                        datetime              time=0,            // heure du point
                        double                price=0,           // prix du point
                        const double          angle=45.0,        // angle de pente
                        const color           clr=clrRed,        // couleur de la ligr
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // style de la ligne
                        const int             width=1,           // largeur de la ligr
                        const bool            back=false,        // en arrière plan
                        const bool            selection=true,    // mise en surbrillar
                        const bool            ray_left=false,    // Prolongement de la
                        const bool            ray_right=true,    // Prolongement de la
                        const bool            hidden=true,       // caché dans la list
                        const long            z_order=0)         // priorité pour le c
  {
//--- crée le second point pour faciliter le déplacement de la ligne de tendance avec
   datetime time2=0;
   double   price2=0;
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeTrendEmptyPoints(time,price,time2,price2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée une ligne de tendance en utilisant 2 points
   if(!ObjectCreate(chart_ID,name,OBJ_TRENDBYANGLE,sub_window,time,price,time2,price2)
     {
      Print(__FUNCTION__,
            " : impossible de créer une ligne de tendance ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- change l'angle de la pente de la ligne de tendance ; lors du changement d'angle,
//--- point de la ligne sont redéfinies automatiquement suivant la nouvelle valeur de
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- définit la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit la largeur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de la ligne à la souri
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage de la l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage de la l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie les coordonnées du point d'ancrage de la ligne de tendance
//+------------------------------------------------------------------------------------
bool TrendPointChange(const long   chart_ID=0,       // identifiant du graphique
                      const string name="TrendLine", // nom de la ligne
                      datetime     time=0,           // coordonnées du point d'ancrage
                      double       price=0)          // coordonnées du point d'ancrage
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage de la ligne de tendance
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change l'angle de pente de la ligne de tendance
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
bool TrendAngleChange(const long   chart_ID=0,       // identifiant du graphique
                      const string name="TrendLine", // nom de la ligne de tendance
                      const double angle=45)         // angle de pente de la ligne de
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change l'angle de la pente de la ligne de tendance
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            " : échec de changement de l'angle de pente de la ligne ! Code d'erreur =
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime la ligne de tendance
//+------------------------------------------------------------------------------------
bool TrendDelete(const long   chart_ID=0,       // identifiant du graphique
                 const string name="TrendLine") // nom de la ligne
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime une ligne de tendance
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : impossible de supprimer la ligne de tendance ! Code d'erreur = ",GetLa
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de la ligne de tendance et définit la val
//| par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- si l'heure du premier point n'est pas définie, elle sera définie sur la barre co
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- définit les coordonnées du second point auxiliaire
//--- le deuxième point sera 9 barres plus à gauche et aura le même prix
```

© 2000-2024, MetaQuotes Ltd.

```
   datetime second_point_time[10];
   CopyTime(Symbol(),Period(),time1,10,second_point_time);
   time2=second_point_time[0];
   price2=price1;
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage des ligr
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner la ligne
   int d1=InpDate1*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- crée une ligne de tendance
   if(!TrendByAngleCreate(0,InpName,0,date[d1],price[p1],InpAngle,InpColor,InpStyle,
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
```

```
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace et tourne la ligne
//--- compteur de boucle
   int v_steps=accuracy/2;
//--- déplace le point d'ancrage et change l'angle de pente de la ligne
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1>1)
         p1-=1;
      //--- déplace le point
      if(!TrendPointChange(0,InpName,date[d1],price[p1]))
         return;
      if(!TrendAngleChange(0,InpName,18*(i+1)))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime du graphique
   TrendDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_CYCLES

Lignes de cycles.

Note

La distance entre les lignes est définie par les coordonnées des deux points d'ancrage de l'objet.

Exemple

Le script suivant crée et déplace des lignes de cycle sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée des lignes de cycle sur le graphique."
#property description "Les coordonnées des points d'ancrage sont défini en pourcentage
#property description "pourcentage de la taille de la fenêtre du graphique"
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="Cycles";    // nom de l'objet
input int             InpDate1=10;         // Date du premier point, %
input int             InpPrice1=45;        // Prix du premier point, %
input int             InpDate2=20;         // Date du second point, %
input int             InpPrice2=55;        // Prix du second point, %
input color           InpColor=clrRed;     // Couleur des lignes de cycle
input ENUM_LINE_STYLE InpStyle=STYLE_DOT;  // Style des lignes de cycle
input int             InpWidth=1;          // Largeur des lignes de cycle
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpBack=false;       // Objet en arrière plan
input bool            InpSelection=true;   // mise en surbrillance pour le déplacement
input bool            InpHidden=true;      // caché dans la liste des objets
input long            InpZOrder=0;         // priorité pour le clic de souris
//+------------------------------------------------------------------------------------
//| Crée les lignes de cycle
//+------------------------------------------------------------------------------------
bool CyclesCreate(const long            chart_ID=0,         // identifiant du graphique
                  const string          name="Cycles",      // nom de l'objet
                  const int             sub_window=0,       // indice de sous-fenêtre
                  datetime              time1=0,            // heure du premier point
                  double                price1=0,           // prix du premier point
                  datetime              time2=0,            // heure du deuxième point
                  double                price2=0,           // prix du deuxième point
                  const color           clr=clrRed,         // couleur des lignes de cy
                  const ENUM_LINE_STYLE style=STYLE_SOLID,  // style des lignes de cycl
                  const int             width=1,            // largeur des lignes de cy
                  const bool            back=false,         // en arrière-plan
                  const bool            selection=true,     // mise en surbrillance pou
                  const bool            hidden=true,        // caché dans la liste des
                  const long            z_order=0)          // priorité du clic de sour
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeCyclesEmptyPoints(time1,price1,time2,price2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée les lignes de cycle aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_CYCLES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            " : échec de création des lignes de cycle ! Code d'erreur = ",GetLastError
      return(false);
     }
//--- définit la couleur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style d'affichage des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement des lignes à la souris
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+-----------------------------------------------------------------------------------
bool CyclesPointChange(const long   chart_ID=0,    // identifiant du graphique
                       const string name="Cycles", // nom de l'objet
                       const int    point_index=0, // indice du point d'ancrage
                       datetime     time=0,        // coordonnées du point d'ancrage t
                       double       price=0)       // coordonnées du point d'ancrage p
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Supprime les lignes de cycle
//+-----------------------------------------------------------------------------------
bool CyclesDelete(const long   chart_ID=0,    // identifiant du graphique
                  const string name="Cycles") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime les lignes de cycle
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de suppression des lignes de cycle ! Code d'erreur = ",GetLastEr
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+-----------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage des lignes de cycle et définit la valeur
//| par défaut des points vides
//+-----------------------------------------------------------------------------------
void ChangeCyclesEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
  {
//--- si l'heure du premier point n'est pas définie, elle sera définie sur la barre co
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du second point n'est pas réglée, elle se trouve à 9 barres de la sec
   if(!time2)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- Positionne le deuxième point à 9 barres à gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas défini, il est égal à celui du premier po
   if(!price2)
      price2=price1;
  }
//+-----------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+-----------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage des lignes de cycle
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner les lignes de cycle
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crée une ligne de tendance
   if(!CyclesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage
//--- compteur de boucle
   int h_steps=bars/5;
//--- déplace le second point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d2<bars-1)
         d2+=1;
      //--- déplace le point
      if(!CyclesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   h_steps=bars/4;
//--- déplace le premier point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d1<bars-1)
         d1+=1;
      //--- déplace le point
      if(!CyclesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'objet du graphique
   CyclesDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROWED_LINE

Ligne flêchée.

Exemple

Le script suivant crée et déplace la ligne flêchée sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un objet graphique \"Ligne flêchée\" objet gr
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="ArrowedLine"; // Nom de la ligne
input int             InpDate1=35;           // date du 1er point, %
input int             InpPrice1=60;          // prix du 1er point, %
input int             InpDate2=65;           // date du 2ème point, %
input int             InpPrice2=40;          // prix du 2ème point, %
input color           InpColor=clrRed;       // Couleur de la ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;   // Style de la ligne
input int             InpWidth=2;            // Largeur de la ligne
input bool            InpBack=false;         // Ligne d'arrière plan
input bool            InpSelection=true;     // Mise en surbrillance pour déplacer
input bool            InpHidden=true;        // Caché dans la liste des objets
input long            InpZOrder=0;           // Priorité du clic de souris
```

© 2000-2024, MetaQuotes Ltd.

```
//+-----------------------------------------------------------------------------------
//| Crée une ligne flêchée aux coordonnées données
//+-----------------------------------------------------------------------------------
bool ArrowedLineCreate(const long            chart_ID=0,         // identifiant du gra
                       const string          name="ArrowedLine", // nom de la ligne
                       const int             sub_window=0,       // indice de sous-fen
                       datetime              time1=0,            // heure du 1er point
                       double                price1=0,           // prix du 1er point
                       datetime              time2=0,            // heure du 2nd point
                       double                price2=0,           // prix du 2nd point
                       const color           clr=clrRed,         // couleur de la ligr
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // style de la ligne
                       const int             width=1,            // largeur de la ligr
                       const bool            back=false,         // en arrière-plan
                       const bool            selection=true,     // mise en surbrillar
                       const bool            hidden=true,        // caché dans la list
                       const long            z_order=0)          // priorité du clic c
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowedLineEmptyPoints(time1,price1,time2,price2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée une ligne flêchée aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_ARROWED_LINE,sub_window,time1,price1,time2,price
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer une ligne flêchée ! Code d'erreur = ",GetLastEr
      return(false);
     }
//--- définit la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style d'affichage de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de la ligne à la souri
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+-----------------------------------------------------------------------------------
//| Déplace le point d'ancrage de la ligne flêchée
//+-----------------------------------------------------------------------------------
bool ArrowedLinePointChange(const long   chart_ID=0,         // identifiant du graphic
                            const string name="ArrowedLine", // nom de la ligne
                            const int    point_index=0,      // indice du point d'anc
                            datetime     time=0,             // coordonnées du point c
                            double       price=0)            // coordonnées du point c
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplacez le point d'ancrage de la ligne
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| La fonction supprime la ligne flêchée du graphique
//+-----------------------------------------------------------------------------------
bool ArrowedLineDelete(const long   chart_ID=0,         // identifiant du graphique
                       const string name="ArrowedLine") // nom de la ligne
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime une ligne flêchée
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer une ligne flêchée ! Code d'erreur = ",GetLastEr
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+-----------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
void ChangeArrowedLineEmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2)
  {
//--- si l'heure du premier point n'est pas définie, elle sera définie sur la barre co
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du second point n'est pas réglée, elle se trouve à 9 barres de la sec
   if(!time2)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- Positionne le deuxième point à 9 barres à gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas défini, il est égal à celui du premier po
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage des ligr
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
```

```
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner la ligne
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crée une ligne flêchée
   if(!ArrowedLineCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplacez les points d'ancrage de la ligne
//--- compteur de boucle
   int v_steps=accuracy/5;
//--- déplacez le second point d'ancrage à la verticale
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p2<accuracy-1)
         p2+=1;
      //--- déplace le point
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- déplace le premier point d'ancrage verticalement
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1>1)
         p1-=1;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- déplace le point
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- compteur de boucle
   int h_steps=bars/2;
//--- déplacez les deux points d'ancrage horizontalement en même temps
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise les valeurs suivantes
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- décale les points
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0.03 secondes
      Sleep(30);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime une ligne flêchée
   ArrowedLineDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_CHANNEL

Canal équidistant

Note

Pour un canal équidistant, il est possible de spécifier le mode de prolongement vers la droite et/ou vers la gauche (propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT respectivement). Le mode de remplissage du canal avec une couleur peut également être défini.

Exemple

Le script suivant crée et déplace un canal équidistant sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un objet graphique \"Canal Equidistant\"."
#property description "Les coordonnées du point d'ancrage sont en pourcentage de la ta
#property description "de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
nom du canalinput string          InpName="Channel";   // Nom du canal
input int             InpDate1=25;         // Date du premier point, %
input int             InpPrice1=60;        // Prix du premier point, %
input int             InpDate2=65;         // Date du second point, %
input int             InpPrice2=80;        // Prix du second point, %
input int             InpDate3=30;         // Date du troisième point, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpPrice3=40;        // Prix du troisième point, %
input color           InpColor=clrRed;     // Couleur du canal
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Style des lignes du canal
input int             InpWidth=2;          // Largeur des lignes du canal
input bool            InpBack=false;       // Canal en arrière plan
input bool            InpFill=false;       // Remplissage du canal avec une couleur
input bool            InpSelection=true;   // Mise en surbrillance pour déplacer
input bool            InpRayLeft=false;    // Prolongation du canal vers la gauche
input bool            InpRayRight=false;   // Prolongation du canal vers la droite
input bool            InpHidden=true;      // Caché dans la liste des objets
input long            InpZOrder=0;         // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée un canal équidistant aux coordonnées données
//+------------------------------------------------------------------------------------
bool ChannelCreate(const long            chart_ID=0,        // identifiant du graphiqu
                   const string          name="Channel",    // nom du canal
                   const int             sub_window=0,      // indice de sous-fenêtre
                   datetime              time1=0,           // heure du premier point
                   double                price1=0,          // prix du premier point
                   datetime              time2=0,           // heure du deuxième point
                   double                price2=0,          // prix du deuxième point
                   datetime              time3=0,           // heure du troisième poir
                   double                price3=0,          // prix du troisième point
                   const color           clr=clrRed,        // Couleur du canal
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // Style des lignes du car
                   const int             width=1,           // largeur des lignes du c
                   const bool            fill=false,        // Remplissage du canal av
                   const bool            back=false,        // en arrière plan
                   const bool            selection=true,    // mise en surbrillance po
                   const bool            ray_left=false,    // Prolongation du canal v
                   const bool            ray_right=false,   // Prolongation du canal v
                   const bool            hidden=true,       // caché dans la liste des
                   const long            z_order=0)         // priorité pour le clic c
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée un canal aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_CHANNEL,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": échec de création d'un canal équidistant ! Code d'erreur = ",GetLastEr
      return(false);
     }
//--- définit la couleur du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit la largeur des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- active (true) ou désactive (false) le mode de remplissage du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du canal pour
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage du cana
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage du cana
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage du canal
//+------------------------------------------------------------------------------------
bool ChannelPointChange(const long   chart_ID=0,      // identifiant du graphique
                        const string name="Channel", // nom du canal
                        const int    point_index=0,  // indice du point d'ancrage
                        datetime     time=0,         // coordonnées du point d'ancrage
                        double       price=0)        // coordonnées du point d'ancrage
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+-----------------------------------------------------------------------------------
//| Supprime le canal
//+-----------------------------------------------------------------------------------
bool ChannelDelete(const long   chart_ID=0,     // identifiant du graphique
                   const string name="Channel") // nom du canal
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime le canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de la suppression du canal ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage du canal et définit les valeurs par défau
//| pour les vides
//+-----------------------------------------------------------------------------------
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- si l'heure de deuxième (droite) point n'est pas définie, il sera sur la barre co
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas défini, il aura la valeur Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du premier (gauche) point n'est pas définie, il est positionné 9 barr
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas défini, le déplace 300 points plus haut qu
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si l'heure du premier point n'est pas définie, il sera mis sur la barre courante
   if(!time3)
      time3=time1;
//--- si le prix du troisième point n'est pas défini, il est égal à celui du deuxième
   if(!price3)
      price3=price2;
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage du canal
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner le canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crée le canal équidistant
   if(!ChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
```

© 2000-2024, MetaQuotes Ltd.

```
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage du canal
//--- compteur de boucle
   int h_steps=bars/6;
//--- déplace le second point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d2<bars-1)
         d2+=1;
      //--- déplace le point
      if(!ChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- déplace le premier point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d1>1)
         d1-=1;
      //--- déplace le point
      if(!ChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
```

© 2000-2024, MetaQuotes Ltd.

```
   int v_steps=accuracy/10;
//--- déplace le troisième point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p3>1)
         p3-=1;
      //--- déplace le point
      if(!ChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime le canal du graphique
   ChannelDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_STDDEVCHANNEL

Canal de Déviation Standard.

Note

Pour le Canal de Déviation Standard, il est possible de spécifier le mode de prolongement de son affichage vers la droite et/ou vers la gauche (propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT respectivement). Le mode de remplissage du canal avec une couleur peut également être défini.

La propriété OBJPROP_DEVIATION est utilisée pour changer la valeur de déviation du canal.

Exemple

Le script suivant crée et déplace un Canal de Déviation Standard sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un objet graphique \"Canal de Déviation Stanc
#property description "Les coordonnées du point d'ancrage sont en pourcentage de la ta
#property description "de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="StdDevChannel";    // nom du canal
input int             InpDate1=10;                // Date du premier point, %
input int             InpDate2=40;                // Date du second point, %
input double          InpDeviation=1.0;           // Déviation
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // Couleur du canal
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style des lignes du canal
input int             InpWidth=2;                // largeur des lignes du canal
input bool            InpFill=false;             // Remplissage du canal avec une coul
input bool            InpBack=false;             // Canal en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplacer
input bool            InpRayLeft=false;          // Prolongation du canal vers la gauc
input bool            InpRayRight=false;         // Prolongation du canal vers la droi
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée un Canal de Déviation Standard aux coordonnées données
//+------------------------------------------------------------------------------------
bool StdDevChannelCreate(const long            chart_ID=0,        // identifiant du gr
                         const string          name="Channel",    // nom du canal
                         const int             sub_window=0,      // indice de sous-fe
                         datetime              time1=0,           // heure du premier
                         datetime              time2=0,           // heure du second p
                         const double          deviation=1.0,     // déviation
                         const color           clr=clrRed,        // Couleur du canal
                         const ENUM_LINE_STYLE style=STYLE_SOLID, // Style des lignes
                         const int             width=1,           // largeur des ligne
                         const bool            fill=false,        // Remplissage du ca
                         const bool            back=false,        // en arrière-plan
                         const bool            selection=true,    // mise en surbrilla
                         const bool            ray_left=false,    // Prolongation du c
                         const bool            ray_right=false,   // Prolongation du c
                         const bool            hidden=true,       // caché dans la lis
                         const long            z_order=0)         // priorité du clic
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeChannelEmptyPoints(time1,time2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée un canal aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_STDDEVCHANNEL,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            " : échec de création du canal de déviation standard ! Code d'erreur = ",G
      return(false);
     }
//--- définit la valeur de la déviation affectant la largeur du canal
   ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation);
//--- définit la couleur du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- active (true) ou désactive (false) le mode de remplissage du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du canal pour
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage du cana
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage du cana
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage du canal
//+------------------------------------------------------------------------------------
bool StdDevChannelPointChange(const long   chart_ID=0,     // identifiant du graphique
                              const string name="Channel", // nom du canal
                              const int    point_index=0,  // indice du point d'ancrag
                              datetime     time=0)         // coordonnées du point d'a
  {
//--- si l'heure du point n'est pas définie, déplace le point sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change la déviation du canal
//+------------------------------------------------------------------------------------
bool StdDevChannelDeviationChange(const long   chart_ID=0,     // identifiant du graph
                                  const string name="Channel", // nom du canal
```

```
                                    const double deviation=1.0)  // déviation
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change l'angle de la pente de la ligne de tendance
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation))
     {
      Print(__FUNCTION__,
            " : échec du changement de déviation du canal ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le canal
//+------------------------------------------------------------------------------------
bool StdDevChannelDelete(const long   chart_ID=0,     // identifiant du graphique
                         const string name="Channel") // nom du canal
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime le canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de la suppression du canal ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage du canal et définit les valeurs par défau
//| pour les vides
//+------------------------------------------------------------------------------------
void ChangeChannelEmptyPoints(datetime &time1,datetime &time2)
  {
//--- si l'heure du deuxième point n'est pas définie, il sera mis sur la barre courant
   if(!time2)
      time2=TimeCurrent();
//--- si l'heure du premier point n'est pas réglée, il est mis 9 barres à gauche du de
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
```

```
      }
   }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
   {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
      {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
      }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage du canal
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
      {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
      }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner le canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- crée le canal de déviation standard
   if(!StdDevChannelCreate(0,InpName,0,date[d1],date[d2],InpDeviation,InpColor,InpStyl
      InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder
      {
      return;
      }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant le canal horizontalement vers la droite et l'étend
//--- compteur de boucle
   int h_steps=bars/2;
//--- déplace le canal
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise les valeurs suivantes
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- déplace les points d'ancrage
      if(!StdDevChannelPointChange(0,InpName,0,date[d1]))
         return;
      if(!StdDevChannelPointChange(0,InpName,1,date[d2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   double v_steps=InpDeviation*2;
//--- étend le canal
   for(double i=InpDeviation;i<v_steps;i+=10.0/accuracy)
     {
      if(!StdDevChannelDeviationChange(0,InpName,i))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime le canal du graphique
   StdDevChannelDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
```

© 2000-2024, MetaQuotes Ltd.

```
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_REGRESSION

Canal de Régression Linéaire.

Note

Pour le Canal de Régression Linéaire, il est possible de spécifier le mode de prolongement de son affichage vers la droite et/ou vers la gauche (propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT respectivement). Le mode de remplissage du canal avec une couleur peut également être défini.

Exemple

Le script suivant crée et déplace un Canal de Régression Linéaire sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un objet graphique \"Canal de Régression Lin
#property description "Les coordonnées du point d'ancrage sont en pourcentage de la ta
#property description "de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="Regression"; // nom du canal
input int             InpDate1=10;          // Date du premier point, %
input int             InpDate2=40;          // Date du second point, %
input color           InpColor=clrRed;      // Couleur du canal
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Style des lignes du canal
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpWidth=2;           // largeur des lignes du canal
input bool            InpFill=false;        // Remplissage du canal avec une couleur
input bool            InpBack=false;        // Canal en arrière plan
input bool            InpSelection=true;    // Mise en surbrillance pour déplacer
input bool            InpRayLeft=false;     // Prolongation du canal vers la gauche
input bool            InpRayRight=false;    // Prolongation du canal vers la droite
input bool            InpHidden=true;       // Caché de la liste des objets
input long            InpZOrder=0;          // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée un Canal de Régression Linéaire aux coordonnées données
//+------------------------------------------------------------------------------------
bool RegressionCreate(const long            chart_ID=0,        // identifiant du graph
                      const string          name="Regression", // nom du canal
                      const int             sub_window=0,      // indice de sous-fenêt
                      datetime              time1=0,           // heure du premier poi
                      datetime              time2=0,           // heure du second poir
                      const color           clr=clrRed,        // Couleur du canal
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // Style des lignes du
                      const int             width=1,           // largeur des lignes c
                      const bool            fill=false,        // Remplissage du canal
                      const bool            back=false,        // en arrière-plan
                      const bool            selection=true,    // mise ne surbrillance
                      const bool            ray_left=false,    // Prolongation du cana
                      const bool            ray_right=false,   // Prolongation du cana
                      const bool            hidden=true,       // caché dans la liste
                      const long            z_order=0)         // priorité du clic de
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeRegressionEmptyPoints(time1,time2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée un canal aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_REGRESSION,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            " : échec de création d'un canal de régression linéaire ! Code d'erreur =
      return(false);
     }
//--- définit la couleur du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- active (true) ou désactive (false) le mode de remplissage du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du canal pour
```

© 2000-2024, MetaQuotes Ltd.

```
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage du cana
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage du cana
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage du canal
//+------------------------------------------------------------------------------------
bool RegressionPointChange(const long   chart_ID=0,      // identifiant du graphique
                           const string name="Channel", // nom du canal
                           const int    point_index=0,  // indice du point d'ancrage
                           datetime     time=0)         // coordonnées du point d'ancr
  {
//--- si l'heure du point n'est pas définie, déplace le point sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le canal
//+------------------------------------------------------------------------------------
bool RegressionDelete(const long   chart_ID=0,      // identifiant du graphique
                      const string name="Channel") // nom du canal
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime le canal
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             " : échec de la suppression du canal ! Code d'erreur = ",GetLastError());
       return(false);
      }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage du canal et définit les valeurs par défau
//| pour les vides
//+------------------------------------------------------------------------------------
void ChangeRegressionEmptyPoints(datetime &time1,datetime &time2)
  {
//--- si l'heure du deuxième point n'est pas définie, il sera mis sur la barre courant
   if(!time2)
      time2=TimeCurrent();
//--- si l'heure du premier point n'est pas réglée, il est mis 9 barres à gauche du de
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage du canal
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
```

© 2000-2024, MetaQuotes Ltd.

```
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner le canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- crée le canal de régression linéaire
   if(!RegressionCreate(0,InpName,0,date[d1],date[d2],InpColor,InpStyle,InpWidth,
      InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant le canal horizontalement vers la droite
//--- compteur de boucle
   int h_steps=bars/2;
//--- déplace le canal
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise les valeurs suivantes
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- déplace les points d'ancrage
      if(!RegressionPointChange(0,InpName,0,date[d1]))
         return;
      if(!RegressionPointChange(0,InpName,1,date[d2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime le canal du graphique
   RegressionDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_PITCHFORK

Fourchette d'Andrews

Note

Pour la Fourchette d'Andrews, il est possible de spécifier le mode de prolongement de son affichage vers la droite et/ou vers la gauche (propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT respectivement).

Vous pouvez également spécifier le nombre de lignes-niveaux, leurs valeurs et leurs couleurs.

Exemple

Le script suivant crée et déplace une Fourchette d'Andrews sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Fourchette d'Andrews\"."
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="Pitchfork";  // Nom de la fourchette
input int             InpDate1=14;          // Date du premier point, %
input int             InpPrice1=40;         // Prix du premier point, %
input int             InpDate2=18;          // Date du second point, %
input int             InpPrice2=50;         // Prix du second point, %
input int             InpDate3=18;          // Date du troisième point, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpPrice3=30;         // Prix du troisième point, %
input color           InpColor=clrRed;      // Couleur de la fourchette
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Style de ligne de la fourchette
input int             InpWidth=1;           // Largeur de ligne de la fourchette
input bool            InpBack=false;        // Arrière-plan de la fourchette
input bool            InpSelection=true;    // Mise en surbrillance pour déplacer
input bool            InpRayLeft=false;     // Prolongement de la Fourchette vers la g
input bool            InpRayRight=false;    // Prolongement de la Fourchette vers la d
input bool            InpHidden=true;       // Caché de la liste des objets
input long            InpZOrder=0;          // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée la Fourchette d'Andrews aux coordonnées données
//+------------------------------------------------------------------------------------
bool PitchforkCreate(const long            chart_ID=0,        // identifiant du graphi
                     const string          name="Pitchfork",  // nom de la fourchette
                     const int             sub_window=0,      // indice de sous-fenêtr
                     datetime              time1=0,           // heure du premier poin
                     double                price1=0,          // prix du premier point
                     datetime              time2=0,           // heure du second point
                     double                price2=0,          // prix du second point
                     datetime              time3=0,           // heure du troisième po
                     double                price3=0,          // prix du troisième poi
                     const color           clr=clrRed,        // couleur des lignes de
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // style des lignes de l
                     const int             width=1,           // largeur des lignes de
                     const bool            back=false,        // en arrière-plan
                     const bool            selection=true,    // mise en surbrillance
                     const bool            ray_left=false,    // Prolongement de la Fo
                     const bool            ray_right=false,   // Prolongement de la Fo
                     const bool            hidden=true,       // caché dans la liste d
                     const long            z_order=0)         // priorité du clic de s
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée la Fourchette d'Andrews par les coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_PITCHFORK,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": impossible de créer \"Fourchette d'Andrews\"! Code d'erreur = ",GetLast
      return(false);
     }
//--- définit la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (vrai) ou désactive (faux) le mode de mise en surbrillance de la fourchet
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage de la f
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage de la f
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Définit le nombre de niveaux de la Fourchette d'Andrews et leurs paramètres
//+------------------------------------------------------------------------------------
bool PitchforkLevelsSet(int             levels,           // nombre de lignes de nivea
                        double          &values[],        // valeurs des lignes de niv
                        color           &colors[],        // couleur des lignes de niv
                        ENUM_LINE_STYLE &styles[],        // style des lignes de nivea
                        int             &widths[],        // largeur des lignes de niv
                        const long      chart_ID=0,       // identifiant du graphique
                        const string    name="Pitchfork") // nom de la fourchette
  {
//--- vérifie les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__," : la taille du tableau ne correspond pas au nombre de nivea
      return(false);
     }
//--- définit le nombre de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- définit les propriétés des niveaux dans la boucle
   for(int i=0;i<levels;i++)
     {
      //--- valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largeur du niveau
```

© 2000-2024, MetaQuotes Ltd.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage de la Fourchette d'Andrews
//+------------------------------------------------------------------------------------
bool PitchforkPointChange(const long   chart_ID=0,       // identifiant du graphique
                          const string name="Pitchfork", // nom du canal
                          const int    point_index=0,    // indice du point d'ancrage
                          datetime     time=0,           // coordonnée horaire du poir
                          double       price=0)          // coordonnée prix du point d
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime la Fourchette d'Andrews
//+------------------------------------------------------------------------------------
bool PitchforkDelete(const long   chart_ID=0,       // identifiant du graphique
                     const string name="Pitchfork") // nom du canal
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime le canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : impossible de supprimer la \"Fourchette d'Andrews\"! Code d'erreur = "
      return(false);
     }
//--- exécution réussie
```

© 2000-2024, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de la Fourchette d'Andrews et définit la
//| par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- si l'heure du deuxième (supérieur droit) point n'est pas définie, il sera sur la
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas défini, il aura la valeur Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du premier (gauche) point n'est pas définie, il est positionné 9 bar
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas défini, le déplace 200 points plus bas que
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si l'heure du troisième point n'est pas définie, il coincidera avec celle du deu
   if(!time3)
      time3=time2;
//--- si le prix du troisième point n'est pas défini, le déplace de 200 points plus ba
   if(!price3)
      price3=price1-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage de la Fo
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner la Fourchette d'Andrews
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crée la fourchette
   if(!PitchforkCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pric
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplace les points d'ancrage de la fourchette
//--- compteur de boucle
   int v_steps=accuracy/10;
//--- déplace le premier point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1>1)
         p1-=1;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- déplace le point
      if(!PitchforkPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   int h_steps=bars/8;
//--- déplace le troisième point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d3<bars-1)
         d3+=1;
      //--- déplace le point
      if(!PitchforkPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   v_steps=accuracy/10;
//--- déplace le second point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p2>1)
         p2-=1;
      //--- déplace le point
      if(!PitchforkPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
```

© 2000-2024, MetaQuotes Ltd.

```
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime la fourchette du graphique
   PitchforkDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_GANNLINE

Ligne de Gann.

Note

Pour la "Ligne de Gann", il est possible de spécifier le mode de prolongement de son affichage vers la droite et/ou vers la gauche (propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT respectivement).

Pour déterminer la pente de la ligne, il est possible d'utiliser un angle de Gann avec une échelle ou les coordonnées du deuxième point d'ancrage.

Exemple

Le script suivant crée et déplace une Ligne de Gann sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Ligne de Gann\"."
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="GannLine";        // Nom de la ligne
input int             InpDate1=20;               // Date du premier point, %
input int             InpPrice1=75;              // Prix du premier point, %
input int             InpDate2=80;               // Date du second point, %
```

© 2000-2024, MetaQuotes Ltd.

```
input double          InpAngle=0.0;              // Angle de Gann
input double          InpScale=1.0;              // Echelle
input color           InpColor=clrRed;           // Couleur de ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style de ligne
input int             InpWidth=2;                // Largeur de ligne
input bool            InpBack=false;             // Ligne en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplacer
input bool            InpRayLeft=false;          // Prolongement de la ligne vers la g
input bool            InpRayRight=true;          // Prolongement de la ligne vers la d
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée la Ligne de Gann aux coordonnées, angle et échelle donnés
//+------------------------------------------------------------------------------------
bool GannLineCreate(const long            chart_ID=0,        // identifiant du graphic
                    const string          name="GannLine",   // Nom de la ligne
                    const int             sub_window=0,      // indice de sous-fenêtre
                    datetime              time1=0,           // heure du premier point
                    double                price1=0,          // prix du premier point
                    datetime              time2=0,           // heure du deuxième poir
                    const double          angle=1.0,         // Angle de Gann
                    const double          scale=1.0,         // Echelle
                    const color           clr=clrRed,        // Couleur de ligne
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Style de ligne
                    const int             width=1,           // Largeur de ligne
                    const bool            back=false,        // en arrière-plan
                    const bool            selection=true,    // mise en surbrillance p
                    const bool            ray_left=false,    // Prolongement de la lig
                    const bool            ray_right=true,    // Prolongement de la lig
                    const bool            hidden=true,       // caché dans la liste de
                    const long            z_order=0)         // priorité du clic de sc
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeGannLineEmptyPoints(time1,price1,time2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée la Ligne de Gann aux coordonnées données
//--- corrige les coordonnées du deuxième point d'ancrage si elles sont redéfinies
//--- automatiquement après que l'angle de Gann et/ou l'échelle change,
   if(!ObjectCreate(chart_ID,name,OBJ_GANNLINE,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            " : échec de création de la \"Ligne de Gann\" ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- change l'angle de Gann
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- change l'échelle (nombre de pips par barre)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
```

```
//--- définit la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style d'affichage de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance des lignes po
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage de la l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage de la l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage de la Ligne de Gann
//+------------------------------------------------------------------------------------
bool GannLinePointChange(const long   chart_ID=0,       // identifiant du graphique
                         const string name="GannLine",  // Nom de la ligne
                         const int    point_index=0,    // indice du point d'ancrage
                         datetime     time=0,           // coordonnées du point d'ancra
                         double       price=0)          // coordonnées du point d'ancra
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplacez le point d'ancrage de la ligne
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
```

```
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change l'angle de Gann
//+------------------------------------------------------------------------------------
bool GannLineAngleChange(const long   chart_ID=0,      // identifiant du graphique
                         const string name="GannLine", // Nom de la ligne
                         const double angle=1.0)       // Angle de Gann
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change l'angle de Gann
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            " : échec de changement de l'angle de Gann ! Code d'erreur = ",GetLastErro
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change l'échelle de la Ligne de Gann
//+------------------------------------------------------------------------------------
bool GannLineScaleChange(const long   chart_ID=0,      // identifiant du graphique
                         const string name="GannLine", // Nom de la ligne
                         const double scale=1.0)       // Echelle
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change l'échelle (nombre de pips par barre)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            " : échec de changement de l'échelle ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| La fonction supprime la Ligne de Gann du graphique
//+------------------------------------------------------------------------------------
bool GannLineDelete(const long   chart_ID=0,      // identifiant du graphique
                    const string name="GannLine") // Nom de la ligne
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime la ligne de Gann
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de suppression de la \"Ligne de Gann\" ! Code d'erreur = ",GetLa
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de la Ligne de Gann et définit la valeur
//| par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeGannLineEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- si l'heure du deuxième point n'est pas définie, il sera mis sur la barre courant
   if(!time2)
      time2=TimeCurrent();
//--- si l'heure du premier point n'est pas réglée, il est mis 9 barres à gauche du de
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage des ligr
```

© 2000-2024, MetaQuotes Ltd.

```
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner la Ligne de Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- crée la Ligne de Gann
   if(!GannLineCreate(0,InpName,0,date[d1],price[p1],date[d2],InpAngle,InpScale,InpCol
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage de la ligne et change l'angle
//--- compteur de boucle
   int v_steps=accuracy/2;
//--- déplace le premier point d'ancrage verticalement
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1>1)
         p1-=1;
      //--- déplace le point
      if(!GannLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
```

© 2000-2024, MetaQuotes Ltd.

```
      ChartRedraw();
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- définit la valeur actuelle de l'angle de Gann (changé
//--- après le déplacement du premier point d'ancrage)
   double curr_angle;
   if(!ObjectGetDouble(0,InpName,OBJPROP_ANGLE,0,curr_angle))
      return;
//--- compteur de boucle
   v_steps=accuracy/8;
//--- change l'angle de Gann
   for(int i=0;i<v_steps;i++)
     {
      if(!GannLineAngleChange(0,InpName,curr_angle-0.05*i))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- efface la ligne du graphique
   GannLineDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_GANNFAN

Eventail de Gann.

Note

Pour l'Eventail de Gann, il est possible de spécifier le type de tendance avec l'énumération ENUM_GANN_DIRECTION. En ajustant la valeur de l'échelle (OBJPROP_SCALE), il est possible de changer l'angle de pente des lignes de l'éventail.

Exemple

Le script suivant crée et déplace l'Eventail de Gann sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Eventail de Gann\"."
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="GannFan";          // Nom de l'éventail
input int             InpDate1=15;                // Date du premier point, %
input int             InpPrice1=25;               // Prix du premier point, %
input int             InpDate2=85;                // Date du second point, %
input double          InpScale=2.0;               // Echelle
input bool            InpDirection=false;         // Direction de la tendance
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // Couleur de l'éventail
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style des lignes de l'éventail
input int             InpWidth=1;                // Largeur des lignes de l'éventail
input bool            InpBack=false;             // Eventail en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplacer
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée l'Eventail de Gann
//+------------------------------------------------------------------------------------
bool GannFanCreate(const long            chart_ID=0,        // identifiant du graphiqu
                   const string          name="GannFan",    // nom de l'éventail
                   const int             sub_window=0,      // indice de sous-fenêtre
                   datetime              time1=0,           // heure du premier point
                   double                price1=0,          // prix du premier point
                   datetime              time2=0,           // heure du deuxième point
                   const double          scale=1.0,         // Echelle
                   const bool            direction=true,    // direction de la tendanc
                   const color           clr=clrRed,        // couleur de l'éventail
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // style des lignes de l'é
                   const int             width=1,           // largeur des lignes de l
                   const bool            back=false,        // en arrière plan
                   const bool            selection=true,    // mise en surbrillance po
                   const bool            hidden=true,       // caché dans la liste des
                   const long            z_order=0)         // priorité pour le clic d
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeGannFanEmptyPoints(time1,price1,time2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée l'Eventail de Gann aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_GANNFAN,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            " : échec de création de l'\"Eventail de Gann\" ! Code d'erreur = ",GetLas
      return(false);
     }
//--- change l'échelle (nombre de pips par barre)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- change la direction de la tendance de l'Eventail de Gann (true - baissière, fals
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- définit la couleur de l'éventail
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style d'affichage des lignes de l'éventail
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes de l'éventail
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- active (true) ou désactive (false) le mode de mise en surbrillance de l'éventail
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage de l'Eventail de Gann
//+------------------------------------------------------------------------------------
bool GannFanPointChange(const long   chart_ID=0,      // identifiant du graphique
                        const string name="GannFan",  // nom de l'éventail
                        const int    point_index=0,   // indice du point d'ancrage
                        datetime     time=0,          // coordonnées du point d'ancrage
                        double       price=0)         // coordonnées du point d'ancrage
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage de l'éventail
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change l'échelle de l'Eventail de Gann
//+------------------------------------------------------------------------------------
bool GannFanScaleChange(const long   chart_ID=0,      // identifiant du graphique
                        const string name="GannFan",  // nom de l'éventail
                        const double scale=1.0)       // Echelle
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change l'échelle (nombre de pips par barre)
```

```
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            " : échec de changement de l'échelle ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change la direction de la tendance de l'Eventail de Gann
//+------------------------------------------------------------------------------------
bool GannFanDirectionChange(const long   chart_ID=0,     // identifiant du graphique
                            const string name="GannFan", // nom de l'éventail
                            const bool   direction=true) // direction de la tendance
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change la direction de la tendance de l'Eventail de Gann
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            " : échec de changement de l'échelle ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| La fonction supprime l'Eventail de Gann du graphique
//+------------------------------------------------------------------------------------
bool GannFanDelete(const long   chart_ID=0,     // identifiant du graphique
                   const string name="GannFan") // nom de l'éventail
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'Eventail de Gann
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de suppression de l'\"Eventail de Gann\" ! Code d'erreur = ",Get
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de l'Eventail de Gann et définit la valeu
//| par défaut des points vides
```

© 2000-2024, MetaQuotes Ltd.

```
//+-----------------------------------------------------------------------------------
void ChangeGannFanEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- si l'heure du deuxième point n'est pas définie, il sera mis sur la barre courant
   if(!time2)
      time2=TimeCurrent();
//--- si l'heure du premier point n'est pas réglée, il est mis 9 barres à gauche du de
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+-----------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+-----------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage de l'éve
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
```

```
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner l'Eventail de Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- crée l'Eventail de Gann
   if(!GannFanCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage de l'éventail
//--- compteur de boucle
   int v_steps=accuracy/2;
//--- déplace le premier point d'ancrage verticalement
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1<accuracy-1)
         p1+=1;
      //--- déplace le point
      if(!GannFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- change la direction de la tendance de l'éventail en tendance baissière
   GannFanDirectionChange(0,InpName,true);
//--- redessine le graphique
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'éventail du graphique
   GannFanDelete(0,InpName);
```

© 2000-2024, MetaQuotes Ltd.

```
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_GANNGRID

Grille de Gann.

Note

Pour la Grille de Gann, il est possible de spécifier le type de tendance avec l'énumération ENUM_GANN_DIRECTION. En ajustant la valeur de l'échelle (OBJPROP_SCALE), il est possible de changer l'angle de pente des lignes de la grille.

Exemple

Le script suivant crée et déplace la Grille de Gann sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Grille de Gann\"."
#property description "Les coordonnées du point d'ancrage de la grille sont définies e
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="GannGrid";        // Nom de la grille
input int             InpDate1=15;               // Date du premier point, %
input int             InpPrice1=25;              // Prix du premier point, %
input int             InpDate2=35;               // Date du second point, %
input double          InpScale=3.0;              // Echelle
input bool            InpDirection=false;        // Direction de la tendance
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // Couleur de la grille
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style des lignes de la grille
input int             InpWidth=1;                // Largeur des lignes de l'éventail
input bool            InpBack=false;             // Grille en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplacer
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée la Grille de Gann
//+------------------------------------------------------------------------------------
bool GannGridCreate(const long            chart_ID=0,        // identifiant du graphic
                    const string          name="GannGrid",   // Nom de la grille
                    const int             sub_window=0,      // indice de sous-fenêtre
                    datetime              time1=0,           // heure du premier point
                    double                price1=0,          // prix du premier point
                    datetime              time2=0,           // heure du deuxième poir
                    const double          scale=1.0,         // Echelle
                    const bool            direction=true,    // direction de la tendar
                    const color           clr=clrRed,        // Couleur de la grille
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Style des lignes de la
                    const int             width=1,           // largeur des lignes de
                    const bool            back=false,        // en arrière-plan
                    const bool            selection=true,    // mise en surbrillance p
                    const bool            hidden=true,       // caché dans la liste de
                    const long            z_order=0)         // priorité du clic de sc
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeGannGridEmptyPoints(time1,price1,time2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée la Grille de Gann aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_GANNGRID,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            " : échec de création de la \"Grille de Gann\" ! Code d'erreur = ",GetLast
      return(false);
     }
//--- change l'échelle (nombre de pips par barre)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- change la direction de la tendance de l'Eventail de Gann (true - baissière, fals
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- définit la couleur de la grille
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style d'affichage des lignes de la grille
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes de la grille
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- active (true) ou désactive (false) le mode de mise en surbrillance de la grille
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage de la Grille de Gann
//+------------------------------------------------------------------------------------
bool GannGridPointChange(const long   chart_ID=0,      // identifiant du graphique
                         const string name="GannGrid", // Nom de la grille
                         const int    point_index=0,   // indice du point d'ancrage
                         datetime     time=0,          // coordonnées du point d'ancra
                         double       price=0)         // coordonnées du point d'ancra
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage de la grille
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change l'échelle de la Grille de Gann
//+------------------------------------------------------------------------------------
bool GannGridScaleChange(const long   chart_ID=0,      // identifiant du graphique
                         const string name="GannGrid", // grilles
                         const double scale=1.0)       // Echelle
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change l'échelle (nombre de pips par barre)
```

```
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            " : échec de changement de l'échelle ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change la direction de la tendance de la Grille de Gann
//+------------------------------------------------------------------------------------
bool GannGridDirectionChange(const long   chart_ID=0,      // identifiant du graphique
                             const string name="GannGrid", // Nom de la grille
                             const bool   direction=true)  // direction de la tendance
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change la direction de la tendance de la Grille de Gann
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            " : échec de changement de l'échelle ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| La fonction supprime l'Eventail de Gann du graphique
//+------------------------------------------------------------------------------------
bool GannGridDelete(const long   chart_ID=0,      // identifiant du graphique
                    const string name="GannGrid") // Nom de la grille
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime la Grille de Gann
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de suppression de la \"Grille de Gann\" ! Code d'erreur = ",GetL
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de la Grille de Gann et définit la valeur
//| par défaut des points vides
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------------------------
void ChangeGannGridEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- si l'heure du deuxième point n'est pas définie, il sera mis sur la barre courant
   if(!time2)
      time2=TimeCurrent();
//--- si l'heure du premier point n'est pas réglée, il est mis 9 barres à gauche du de
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage de la grille
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
```

```
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner la Grille de Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- crée la Grille de Gann
   if(!GannGridCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage de la grille
//--- compteur de boucle
   int v_steps=accuracy/4;
//--- déplace le premier point d'ancrage verticalement
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1<accuracy-1)
         p1+=1;
      if(!GannGridPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   int h_steps=bars/4;
//--- déplace le deuxième point horizontalement
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d2<bars-1)
         d2+=1;
      if(!GannGridPointChange(0,InpName,1,date[d2],0))
```

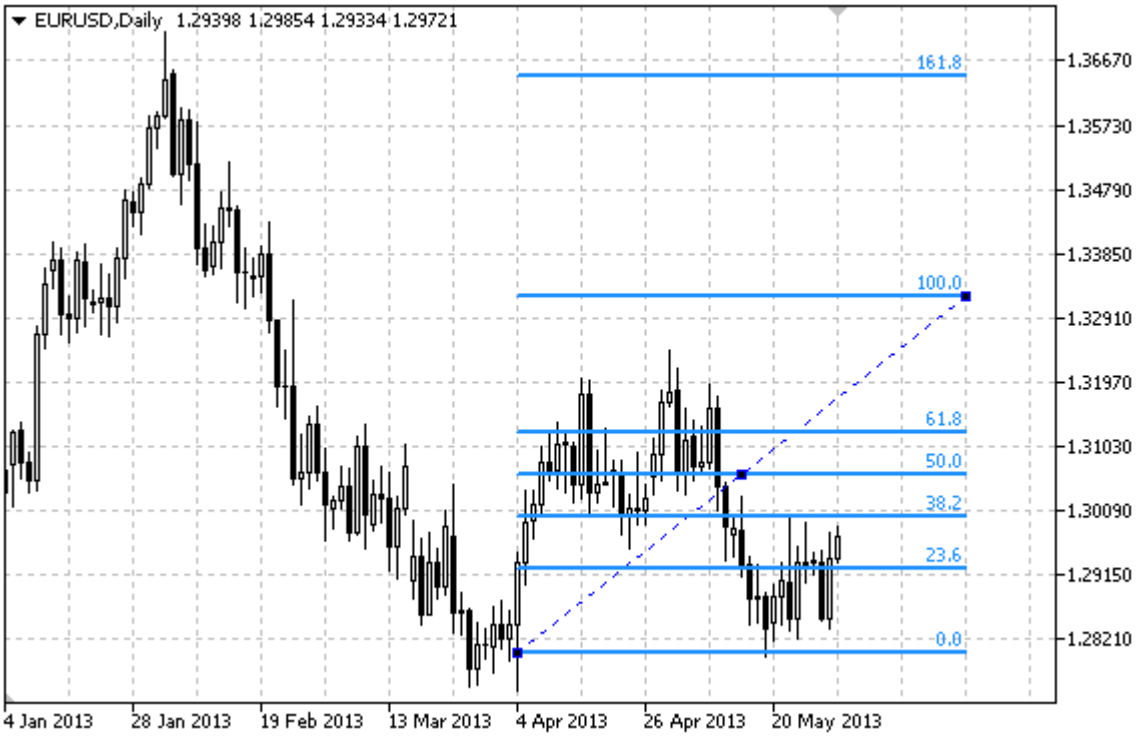
© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- change la direction de la tendance de la grille en tendance baissière
   GannGridDirectionChange(0,InpName,true);
//--- redessine le graphique
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime la grille du graphique
   GannGridDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_FIBO

Retracements de Fibonacci.



Note

Pour les "Retracements de Fibonacci", il est possible de spécifier le mode de prolongement de son affichage vers la droite et/ou vers la gauche (propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT respectivement).

Vous pouvez également spécifier le nombre de lignes-niveaux, leurs valeurs et leurs couleurs.

Exemple

Le script suivant crée et déplace les Retracements de Fibonacci sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Retracements de Fibonacci
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="FiboLevels";       // nom de l'objet
input int             InpDate1=10;                // Date du premier point, %
input int             InpPrice1=65;               // Prix du premier point, %
input int             InpDate2=90;                // Date du second point, %
input int             InpPrice2=85;               // Prix du second point, %
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // couleur de l'objet
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style de ligne
input int             InpWidth=2;                // Largeur de ligne
input bool            InpBack=false;             // Objet en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplacer
input bool            InpRayLeft=false;          // Prolongement de l'objet vers la ga
input bool            InpRayRight=false;         // Prolongement de l'objet vers la di
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée les Retracements de Fibonacci aux coordonnées données
//+------------------------------------------------------------------------------------
bool FiboLevelsCreate(const long            chart_ID=0,        // identifiant du graph
                      const string          name="FiboLevels", // nom de l'objet
                      const int             sub_window=0,      // indice de sous-fenêt
                      datetime              time1=0,           // heure du premier poi
                      double                price1=0,          // prix du premier poin
                      datetime              time2=0,           // heure du second poin
                      double                price2=0,          // prix du deuxième poi
                      const color           clr=clrRed,        // couleur de l'objet
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // Style de ligne de l'
                      const int             width=1,           // largeur de ligne de
                      const bool            back=false,        // en arrière-plan
                      const bool            selection=true,    // mise ne surbrillance
                      const bool            ray_left=false,    // Prolongement de l'ob
                      const bool            ray_right=false,   // Prolongement de l'ob
                      const bool            hidden=true,       // caché dans la liste
                      const long            z_order=0)         // priorité du clic de
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeFiboLevelsEmptyPoints(time1,price1,time2,price2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- Crée l'Expansion de Fibonacci Retracement aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_FIBO,sub_window,time1,price1,time2,price2))a
     {
      Print(__FUNCTION__,
            " : échec de création des \"Retracements de Fibonacci\"! Code d'erreur = "
      return(false);
     }
//--- définit la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du canal pour
```

© 2000-2024, MetaQuotes Ltd.

```
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive (false) le mode de prolongement de l'objet vers la ga
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- active (true) ou désactive (false) le mode de prolongement de l'objet vers la dr
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Définit le nombre de niveaux et leurs paramètres
//+------------------------------------------------------------------------------------
bool FiboLevelsSet(int              levels,            // nombre de niveaux de lignes
                   double           &values[],         // valeurs des niveaux de lignes
                   color            &colors[],         // couleurs des niveaux de ligne
                   ENUM_LINE_STYLE &styles[],          // styles des niveaux de ligne
                   int              &widths[],         // largeur des niveaux de ligne
                   const long       chart_ID=0,        // identifiant du graphique
                   const string     name="FiboLevels") // nom de l'objet
  {
//--- vérifie les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) || levels!=ArraySize(widt
     {
      Print(__FUNCTION__," : la taille du tableau ne correspond pas au nombre de nivea
      return(false);
     }
//--- définit le nombre de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- définit les propriétés des niveaux dans la boucle
   for(int i=0;i<levels;i++)
     {
      //--- valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largeur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Déplace le point d'ancrage des Retracemens de Fibonacci
//+-----------------------------------------------------------------------------------
bool FiboLevelsPointChange(const long   chart_ID=0,        // identifiant du graphique
                           const string name="FiboLevels", // nom de l'objet
                           const int    point_index=0,     // indice du point d'ancrag
                           datetime     time=0,            // coordonnées du point d'a
                           double       price=0)           // coordonnées du point d'a
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Supprime les Retracements de Fibonacci
//+-----------------------------------------------------------------------------------
bool FiboLevelsDelete(const long   chart_ID=0,        // identifiant du graphique
                      const string name="FiboLevels") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de suppression de l'objet \"Retracements de Fibonacci\"! Code d'
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage des Retracements de Fibonacci et définit
```

© 2000-2024, MetaQuotes Ltd.

```
//| par défaut des points vides
//+-----------------------------------------------------------------------------
void ChangeFiboLevelsEmptyPoints(datetime &time1,double &price1,
                                 datetime &time2,double &price2)
  {
//--- si l'heure du deuxième point n'est pas définie, il sera mis sur la barre courant
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas défini, il aura la valeur Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du premier point n'est pas réglée, il est mis 9 barres à gauche du de
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas défini, le déplace 200 points plus bas que
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+-----------------------------------------------------------------------------
//| Fonction de démarrage du script
//+-----------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage des Retr
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
```

```
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner les retracements de Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crée un objet
   if(!FiboLevelsCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage
//--- compteur de boucle
   int v_steps=accuracy*2/5;
//--- déplace le premier point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1>1)
         p1-=1;
      //--- déplace le point
      if(!FiboLevelsPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   v_steps=accuracy*4/5;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- déplace le second point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p2>1)
         p2-=1;
      //--- déplace le point
      if(!FiboLevelsPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'objet du graphique
   FiboLevelsDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_FIBOTIMES

Périodes de Fibonacci.

Note

Pour les "Périodes de Fibonacci", il est possible de spécifier le nombre de lignes-niveaux, leurs valeurs et leurs couleurs.

Exemple

Le script suivant crée et déplace les Périodes de Fibonacci sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Périodes de Fibonacci\".'
#property description "Les coordonnées du point d'ancrage sont en pourcentage de la ta
#property description "de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="FiboTimes";        // nom de l'objet
input int             InpDate1=10;                // Date du premier point, %
input int             InpPrice1=45;               // Prix du premier point, %
input int             InpDate2=20;                // Date du second point, %
input int             InpPrice2=55;               // Prix du second point, %
input color           InpColor=clrRed;            // couleur de l'objet
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style de ligne
input int             InpWidth=2;                 // Largeur de ligne
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpBack=false;             // Objet en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplacer
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée les Périodes de Fibonacci aux coordonnées données
//+------------------------------------------------------------------------------------
bool FiboTimesCreate(const long            chart_ID=0,        // identifiant du graphi
                     const string          name="FiboTimes",  // nom de l'objet
                     const int             sub_window=0,      // indice de sous-fenêtr
                     datetime              time1=0,           // heure du premier poir
                     double                price1=0,          // prix du premier point
                     datetime              time2=0,           // heure du second point
                     double                price2=0,          // prix du second point
                     const color           clr=clrRed,        // couleur de l'objet
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // Style de ligne de l'o
                     const int             width=1,           // largeur de ligne de l
                     const bool            back=false,        // en arrière-plan
                     const bool            selection=true,    // mise en surbrillance
                     const bool            hidden=true,       // caché dans la liste d
                     const long            z_order=0)         // priorité du clic de s
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeFiboTimesEmptyPoints(time1,price1,time2,price2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée les Périodes de Fibonacci aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOTIMES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            " : échec de création des \"Périodes de Fibonacci\" ! Code d'erreur = ",Ge
      return(false);
     }
//--- définit la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du canal pour
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Définit le nombre de niveaux et leurs paramètres
//+-----------------------------------------------------------------------------------
bool FiboTimesLevelsSet(int             levels,           // nombre de niveaux de ligr
                        double          &values[],        // valeurs des lignes de niv
                        color           &colors[],        // couleur des lignes de niv
                        ENUM_LINE_STYLE &styles[],        // style des lignes de nivea
                        int             &widths[],        // largeur des lignes de niv
                        const long      chart_ID=0,       // identifiant du graphique
                        const string    name="FiboTimes") // nom de l'objet
  {
//--- vérifie les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__," : la taille du tableau ne correspond pas au nombre de nivea
      return(false);
     }
//--- définit le nombre de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- définit les propriétés des niveaux dans la boucle
   for(int i=0;i<levels;i++)
     {
      //--- valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largeur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(values[i],1));
     }
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Déplace le point d'ancrage des Périodes de Fibonacci
//+-----------------------------------------------------------------------------------
bool FiboTimesPointChange(const long   chart_ID=0,       // identifiant du graphique
                          const string name="FiboTimes", // nom de l'objet
                          const int    point_index=0,    // indice du point d'ancrage
                          datetime     time=0,           // coordonnée horaire du poir
```

© 2000-2024, MetaQuotes Ltd.

```
                            double       price=0)          // coordonnée prix du point c
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime les Périodes de Fibonacci
//+------------------------------------------------------------------------------------
bool FiboTimesDelete(const long   chart_ID=0,       // identifiant du graphique
                     const string name="FiboTimes") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de suppression des \"Périodes de Fibonacci\" ! Code d'erreur = '
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage des Périodes de Fibonacci et définit la v
//| et définit les valeurs par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeFiboTimesEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- si l'heure du premier point n'est pas définie, elle sera définie sur la barre co
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
```

```
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du second point n'est pas réglée, il est mis à 9 barres à gauche du s
   if(!time2)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 3 dernières barres
      datetime temp[3];
      CopyTime(Symbol(),Period(),time1,3,temp);
      //--- définit le premier point 2 barres à gauche du deuxième
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas défini, il est égal à celui du premier po
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage des Pér
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
```

© 2000-2024, MetaQuotes Ltd.

```
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner les Périodes de Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crée un objet
   if(!FiboTimesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage
//--- compteur de boucle
   int h_steps=bars*2/5;
//--- déplace le second point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d2<bars-1)
         d2+=1;
      //--- déplace le point
      if(!FiboTimesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   h_steps=bars*3/5;
//--- déplace le premier point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d1<bars-1)
         d1+=1;
      //--- déplace le point
      if(!FiboTimesPointChange(0,InpName,0,date[d1],price[p1]))
```

© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'objet du graphique
   FiboTimesDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_FIBOFAN

Eventail de Fibonacci.

Note

Pour l'"Eventail de Fibonacci", il est possible de spécifier le nombre de lignes-niveaux, leurs valeurs et leurs couleurs.

Exemple

Le script suivant crée et déplace l'Eventail de Fibonacci sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Eventail de Fibonacci\".'
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="FiboFan";          // Nom de l'éventail
input int             InpDate1=10;                // Date du premier point, %
input int             InpPrice1=25;               // Prix du premier point, %
input int             InpDate2=30;                // Date du second point, %
input int             InpPrice2=50;               // Prix du second point, %
input color           InpColor=clrRed;            // Couleur de ligne de l'éventail
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style de ligne
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpWidth=2;                // Largeur de ligne
input bool            InpBack=false;             // Objet en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplacer
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée l'Eventail de Fibonacci aux coordonnées données
//+------------------------------------------------------------------------------------
bool FiboFanCreate(const long            chart_ID=0,        // identifiant du graphiqu
                   const string          name="FiboFan",    // nom de l'éventail
                   const int             sub_window=0,      // indice de sous-fenêtre
                   datetime              time1=0,           // heure du premier point
                   double                price1=0,          // prix du premier point
                   datetime              time2=0,           // heure du deuxième point
                   double                price2=0,          // prix du deuxième point
                   const color           clr=clrRed,        // couleur de ligne de l'é
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // style de ligne de l'éve
                   const int             width=1,           // largeur de ligne de l'é
                   const bool            back=false,        // en arrière plan
                   const bool            selection=true,    // mise en surbrillance po
                   const bool            hidden=true,       // caché dans la liste des
                   const long            z_order=0)         // priorité pour le clic d
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeFiboFanEmptyPoints(time1,price1,time2,price2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée l'Eventail de Fibonacci aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOFAN,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            " : échec de création de l'\"Eventail de Fibonacci\"! Code d'erreur = ",Ge
      return(false);
     }
//--- définit la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance de l'éventail
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Définit le nombre de niveaux et leurs paramètres
//+------------------------------------------------------------------------------------
bool FiboFanLevelsSet(int             levels,         // nombre de niveaux de lignes
                      double          &values[],      // valeurs des niveaux de lignes
                      color           &colors[],      // couleurs des niveaux de ligne
                      ENUM_LINE_STYLE &styles[],      // styles des niveaux de ligne
                      int             &widths[],      // largeur des niveaux de ligne
                      const long      chart_ID=0,     // identifiant du graphique
                      const string    name="FiboFan") // nom de l'éventail
  {
//--- vérifie les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__," : la taille du tableau ne correspond pas au nombre de nivea
      return(false);
     }
//--- définit le nombre de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- définit les propriétés des niveaux dans la boucle
   for(int i=0;i<levels;i++)
     {
      //--- valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largeur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage de l'Eventail de Fibonacci
//+------------------------------------------------------------------------------------
bool FiboFanPointChange(const long   chart_ID=0,     // identifiant du graphique
                        const string name="FiboFan", // nom de l'éventail
                        const int    point_index=0,  // indice du point d'ancrage
```

```
                        datetime     time=0,         // coordonnées du point d'ancrage
                        double       price=0)        // coordonnées du point d'ancrage
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime l'Eventail de Fibonacci
//+------------------------------------------------------------------------------------
bool FiboFanDelete(const long   chart_ID=0,     // identifiant du graphique
                   const string name="FiboFan") // nom de l'éventail
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'éventail
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de suppression de l'objet \"Eventail de Fibonacci\"! Code d'erre
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de l'Eventail de Fibonacci et définit la
//| par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeFiboFanEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- si l'heure du deuxième point n'est pas définie, il sera mis sur la barre courant
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas défini, il aura la valeur Bid
```

```
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du premier point n'est pas réglée, il est mis 9 barres à gauche du de
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas défini, le déplace 200 points plus bas que
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage de l'Eve
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner l'Eventail de Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crée un objet
   if(!FiboFanCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage de l'éventail
//--- compteur de boucle
   int v_steps=accuracy/2;
//--- déplace le premier point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1<accuracy-1)
         p1+=1;
      //--- déplace le point
      if(!FiboFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   int h_steps=bars/4;
//--- déplace le second point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d2<bars-1)
         d2+=1;
      //--- déplace le point
      if(!FiboFanPointChange(0,InpName,1,date[d2],price[p2]))
         return;
```

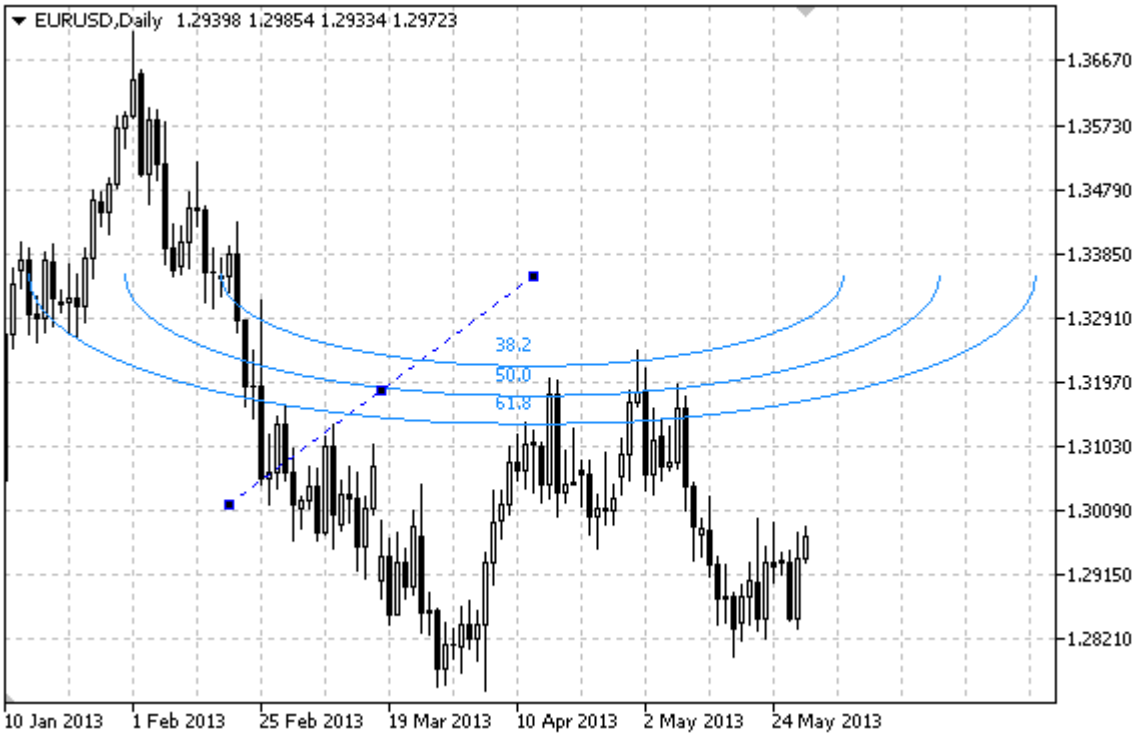
© 2000-2024, MetaQuotes Ltd.

```
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'objet du graphique
   FiboFanDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_FIBOARC

Arcs de Fibonacci.



Note

Pour les "Arcs de Fibonacci", il est possible de spécifier le mode d'affichage de l'ellipse en entier. Le rayon de courbure peut être spécifié en changeant l'échelle et les coordonnées des points d'ancrage.

Vous pouvez également spécifier le nombre de lignes-niveaux, leurs valeurs et leurs couleurs.

Exemple

Le script suivant crée et déplace les Arcs de Fibonacci sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Arcs de Fibonacci\"."
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="FiboArc";          // nom de l'objet
input int             InpDate1=25;                // Date du premier point, %
input int             InpPrice1=25;               // Prix du premier point, %
input int             InpDate2=35;                // Date du second point, %
input int             InpPrice2=55;               // Prix du second point, %
input double          InpScale=3.0;               // Echelle
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpFullEllipse=true;       // Forme des arcs
input color           InpColor=clrRed;           // Couleur de ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style de ligne
input int             InpWidth=2;                // Largeur de ligne
input bool            InpBack=false;             // Objet en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplacer
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée les Arcs de Fibonacci aux coordonnées données
//+------------------------------------------------------------------------------------
bool FiboArcCreate(const long            chart_ID=0,         // identifiant du graphic
                   const string          name="FiboArc",     // nom de l'objet
                   const int             sub_window=0,       // indice de sous-fenêtre
                   datetime              time1=0,            // heure du premier point
                   double                price1=0,           // prix du premier point
                   datetime              time2=0,            // heure du deuxième poin
                   double                price2=0,           // prix du deuxième point
                   const double          scale=1.0,          // Echelle
                   const bool            full_ellipse=false, // forme des arcs
                   const color           clr=clrRed,         // Couleur de ligne
                   const ENUM_LINE_STYLE style=STYLE_SOLID,  // Style de ligne
                   const int             width=1,            // Largeur de ligne
                   const bool            back=false,         // en arrière plan
                   const bool            selection=true,     // mise en surbrillance p
                   const bool            hidden=true,        // caché dans la liste de
                   const long            z_order=0)          // priorité pour le clic
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeFiboArcEmptyPoints(time1,price1,time2,price2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée les Arcs de Fibonacci aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOARC,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            " : échec de création des \"Arcs de Fibonacci\"! Code d'erreur = ",GetLast
      return(false);
     }
//--- définit l'échelle
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- définit l'affichage des arcs comme une ellipse entière (true) ou une demie ellip
   ObjectSetInteger(chart_ID,name,OBJPROP_ELLIPSE,full_ellipse);
//--- définit la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance des arcs pour
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Définit le nombre de niveaux et leurs paramètres
//+------------------------------------------------------------------------------------
bool FiboArcLevelsSet(int             levels,         // nombre de niveaux de lignes
                      double          &values[],      // valeurs des niveaux de lignes
                      color           &colors[],      // couleurs des niveaux de ligne
                      ENUM_LINE_STYLE &styles[],      // styles des niveaux de ligne
                      int             &widths[],      // largeur des niveaux de ligne
                      const long      chart_ID=0,     // identifiant du graphique
                      const string    name="FiboArc") // nom de l'objet
  {
//--- vérifie les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__," : la taille du tableau ne correspond pas au nombre de niveau
      return(false);
     }
//--- définit le nombre de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- définit les propriétés des niveaux dans la boucle
   for(int i=0;i<levels;i++)
     {
      //--- valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largeur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage des Arcs de Fibonacci
//+------------------------------------------------------------------------------------
bool FiboArcPointChange(const long   chart_ID=0,      // identifiant du graphique
                        const string name="FiboArc", // nom de l'objet
                        const int    point_index=0,  // indice du point d'ancrage
                        datetime     time=0,         // coordonnées du point d'ancrage
                        double       price=0)        // coordonnées du point d'ancrage
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime les Arcs de Fibonacci
//+------------------------------------------------------------------------------------
bool FiboArcDelete(const long   chart_ID=0,      // identifiant du graphique
                   const string name="FiboArc") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de suppression des \"Arcs de Fibonacci\"! Code d'erreur = ",GetL
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage des Arcs de Fibonacci et définit la valeu
```

© 2000-2024, MetaQuotes Ltd.

```
//| par défaut des points vides
//+-----------------------------------------------------------------------------
void ChangeFiboArcEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- si l'heure du deuxième point n'est pas définie, il sera mis sur la barre courant
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas défini, il aura la valeur Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du premier point n'est pas réglée, il est mis 9 barres à gauche du de
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas défini, le déplace 300 points plus bas que
   if(!price1)
      price1=price2-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+-----------------------------------------------------------------------------
//| Fonction de démarrage du script
//+-----------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées des points d'ancrage des Arcs
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner les Arcs de Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crée un objet
   if(!FiboArcCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpScale,
      InpFullEllipse,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrd
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage
//--- compteur de boucle
   int v_steps=accuracy/5;
//--- déplace le premier point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1<accuracy-1)
         p1+=1;
      //--- déplace le point
      if(!FiboArcPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   int h_steps=bars/5;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- déplace le second point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d2<bars-1)
         d2+=1;
      //--- déplace le point
      if(!FiboArcPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'objet du graphique
   FiboArcDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_FIBOCHANNEL

Canal de Fibonacci.

Note

Pour le Canal de Fibonacci, il est possible de spécifier le mode de prolongement de son affichage vers la droite et/ou vers la gauche (propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT respectivement).

Vous pouvez également spécifier le nombre de lignes-niveaux, leurs valeurs et leurs couleurs.

Exemple

Le script suivant crée et déplace le Canal de Fibonacci sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Canal de Fibonacci\"."
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="FiboChannel";      // nom du canal
input int             InpDate1=20;                // Date du premier point, %
input int             InpPrice1=10;               // Prix du premier point, %
input int             InpDate2=60;                // Date du second point, %
input int             InpPrice2=30;               // Prix du second point, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpDate3=20;               // Date du troisième point, %
input int             InpPrice3=25;              // Prix du troisième point, %
input color           InpColor=clrRed;           // Couleur du canal
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style des lignes du canal
input int             InpWidth=2;                // largeur des lignes du canal
input bool            InpBack=false;             // Canal en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplace
input bool            InpRayLeft=false;          // Prolongation du canal vers la gau
input bool            InpRayRight=false;         // Prolongation du canal vers la dro
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée le Canal de Fibonacci aux coordonnées données
//+------------------------------------------------------------------------------------
bool FiboChannelCreate(const long            chart_ID=0,         // identifiant du gra
                       const string          name="FiboChannel", // nom du canal
                       const int             sub_window=0,       // indice de sous-fe
                       datetime              time1=0,            // heure du 1er point
                       double                price1=0,           // prix du 1er point
                       datetime              time2=0,            // heure du 2nd point
                       double                price2=0,           // prix du 2nd point
                       datetime              time3=0,            // heure du troisième
                       double                price3=0,           // prix du troisième
                       const color           clr=clrRed,         // Couleur du canal
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // Style des lignes
                       const int             width=1,            // largeur des ligne
                       const bool            back=false,         // en arrière-plan
                       const bool            selection=true,     // mise en surbrilla
                       const bool            ray_left=false,     // Prolongation du c
                       const bool            ray_right=false,    // Prolongation du c
                       const bool            hidden=true,        // caché dans la lis
                       const long            z_order=0)          // priorité du clic
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeFiboChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée un canal aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOCHANNEL,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            " : échec de création du \"Canal de Fibonacci\"! Code d'erreur = ",GetLast
      return(false);
     }
//--- définit la couleur du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes du canal
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du canal pour
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage du cana
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- active (true) ou désactive(false) le mode de prolongation de l'affichage du cana
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Définit le nombre de niveaux et leurs paramètres
//+------------------------------------------------------------------------------------
bool FiboChannelLevelsSet(int             levels,             // nombre de niveaux de
                          double          &values[],          // valeurs des niveaux d
                          color           &colors[],          // couleurs des niveaux
                          ENUM_LINE_STYLE &styles[],          // styles des niveaux de
                          int             &widths[],          // largeur des niveaux d
                          const long      chart_ID=0,         // identifiant du graphi
                          const string    name="FiboChannel") // nom de l'objet
  {
//--- vérifie les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__," : la taille du tableau ne correspond pas au nombre de nivea
      return(false);
     }
//--- définit le nombre de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- définit les propriétés des niveaux dans la boucle
   for(int i=0;i<levels;i++)
     {
      //--- valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- largeur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage du Canal de Fibonacci
//+------------------------------------------------------------------------------------
bool FiboChannelPointChange(const long   chart_ID=0,         // identifiant du graphic
                            const string name="FiboChannel", // nom du canal
                            const int    point_index=0,      // indice du point d'anc
                            datetime     time=0,             // coordonnées du point c
                            double       price=0)            // coordonnées du point c
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le canal
//+------------------------------------------------------------------------------------
bool FiboChannelDelete(const long   chart_ID=0,         // identifiant du graphique
                       const string name="FiboChannel") // nom du canal
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime le canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de suppression du \"Canal de Fibonacci\" ! Code d'erreur = ",Get
      return(false);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage du Canal de Fibonacci et définit la valeu
//| par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeFiboChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                  double &price2,datetime &time3,double &price3)
  {
//--- si l'heure de deuxième (droite) point n'est pas définie, il sera sur la barre co
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas défini, il aura la valeur Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du premier (gauche) point n'est pas définie, il est positionné 9 barr
   if(!time1)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas défini, le déplace 300 points plus haut qu
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si l'heure du premier point n'est pas définie, il sera mis sur la barre courante
   if(!time3)
      time3=time1;
//--- si le prix du troisième point n'est pas défini, il est égal à celui du deuxième
   if(!price3)
      price3=price2;
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
```

© 2000-2024, MetaQuotes Ltd.

```
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage du canal
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner le canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crée le Canal de Fibonacci
   if(!FiboChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],p
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage du canal
//--- compteur de boucle
   int h_steps=bars/10;
//--- déplace le premier point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d1>1)
```

© 2000-2024, MetaQuotes Ltd.

```
         d1-=1;
      //--- déplace le point
      if(!FiboChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   int v_steps=accuracy/10;
//--- déplace le second point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p2>1)
         p2-=1;
      //--- déplace le point
      if(!FiboChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   v_steps=accuracy/15;
//--- déplace le troisième point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p3<accuracy-1)
         p3+=1;
      //--- déplace le point
      if(!FiboChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime le canal du graphique
   FiboChannelDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_EXPANSION

Expansion de Fibonacci.

Note

Pour l'"Expansion de Fibonacci", il est possible de spécifier le mode de prolongement de son affichage vers la droite et/ou vers la gauche (propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT respectivement).

Vous pouvez également spécifier le nombre de lignes-niveaux, leurs valeurs et leurs couleurs.

Exemple

Le script suivant crée et déplace l'Expansion de Fibonacci sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine l'objet graphique \"Expansion de Fibonacci\".
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="FiboExpansion";   // nom de l'objet
input int             InpDate1=10;               // Date du premier point, %
input int             InpPrice1=55;              // Prix du premier point, %
input int             InpDate2=30;               // Date du second point, %
input int             InpPrice2=10;              // Prix du second point, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpDate3=80;               // Date du troisième point, %
input int             InpPrice3=75;              // Prix du troisième point, %
input color           InpColor=clrRed;           // couleur de l'objet
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style des lignes
input int             InpWidth=2;                // Largeur des lignes
input bool            InpBack=false;             // Objet en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplace
input bool            InpRayLeft=false;          // Prolongement de l'objet vers la ga
input bool            InpRayRight=false;         // Prolongement de l'objet vers la di
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée l'Expansion de Fibonacci aux coordonnées données
//+------------------------------------------------------------------------------------
bool FiboExpansionCreate(const long            chart_ID=0,           // identifiant du
                         const string          name="FiboExpansion", // nom du canal
                         const int             sub_window=0,         // indice de sous
                         datetime              time1=0,              // heure du premi
                         double                price1=0,             // prix du premie
                         datetime              time2=0,              // heure du deuxi
                         double                price2=0,             // prix du deuxiè
                         datetime              time3=0,              // heure du trois
                         double                price3=0,             // prix du troisi
                         const color           clr=clrRed,           // couleur de l'o
                         const ENUM_LINE_STYLE style=STYLE_SOLID,    // Style des ligr
                         const int             width=1,              // Largeur des li
                         const bool            back=false,           // en arrière pla
                         const bool            selection=true,       // mise en surbri
                         const bool            ray_left=false,       // Prolongement d
                         const bool            ray_right=false,      // Prolongement d
                         const bool            hidden=true,          // caché dans la
                         const long            z_order=0)            // priorité pour
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeFiboExpansionEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- Crée l'Expansion de Fibonacci aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_EXPANSION,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            " : échec de création de l'objet \"Expansion de Fibonacci\"! Code d'erreur
      return(false);
     }
//--- définit la couleur de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du canal pour
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- active (true) ou désactive (false) le mode de prolongement de l'objet vers la ga
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- active (true) ou désactive (false) le mode de prolongement de l'objet vers la dr
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Définit le nombre de niveaux et leurs paramètres
//+------------------------------------------------------------------------------------
bool FiboExpansionLevelsSet(int              levels,                // nombre de lignes
                            double           &values[],             // valeurs des ligne
                            color            &colors[],             // couleurs des ligr
                            ENUM_LINE_STYLE  &styles[],             // style des lignes
                            int              &widths[],             // largeur des ligne
                            const long       chart_ID=0,            // identifiant du gr
                            const string     name="FiboExpansion") // nom de l'objet
  {
//--- vérifie les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__," : la taille du tableau ne correspond pas au nombre de nivea
      return(false);
     }
//--- définit le nombre de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- définit les propriétés des niveaux dans la boucle
   for(int i=0;i<levels;i++)
     {
      //--- valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- largeur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,"FE "+DoubleToString(100*value
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage de l'Expansion de Fibonacci
//+------------------------------------------------------------------------------------
bool FiboExpansionPointChange(const long   chart_ID=0,           // identifiant du gra
                              const string name="FiboExpansion", // nom de l'objet
                              const int    point_index=0,        // indice du point d'
                              datetime     time=0,               // coordonnées du poi
                              double       price=0)              // coordonnées du poi
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime l'Expansion de Fibonacci
//+------------------------------------------------------------------------------------
bool FiboExpansionDelete(const long   chart_ID=0,           // identifiant du graphiqu
                         const string name="FiboExpansion") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de suppression de l'objet \"Expansion de Fibonacci\"! Code d'err
      return(false);
     }
```

```
//--- exécution réussie
   return(true);
  }
//+-----------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de l'Expansion de Fibonacci et définit la
//| par défaut des points vides
//+-----------------------------------------------------------------------------------
void ChangeFiboExpansionEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                    double &price2,datetime &time3,double &price3)
  {
//--- si l'heure du troisième (droite) point n'est pas définie, il sera mis sur la bar
   if(!time3)
      time3=TimeCurrent();
//--- si le prix du troisième point n'est pas défini, il aura la valeur Bid
   if(!price3)
      price3=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du premier (gauche) point n'est pas définie, il est mis 9 barres à ga
//--- tableau pour récupérer l'heure d'ouverture des 10 dernières barres
   datetime temp[];
   ArrayResize(temp,10);
   if(!time1)
     {
      CopyTime(Symbol(),Period(),time3,10,temp);
      //--- définit le premier point 9 barres à gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas défini, il est égal à celui du troisième p
   if(!price1)
      price1=price3;
//--- si l'heure du deuxième point n'est pas définie, il est mis 7 barres à gauche du
   if(!time2)
      time2=temp[2];
//--- si le prix du deuxième point n'est pas défini, le déplace 250 points plus bas qu
   if(!price2)
      price2=price1-250*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+-----------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+-----------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage de l'objet
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner l'Expansion de Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crée l'Expansion de Fibonacci
   if(!FiboExpansionCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage
//--- compteur de boucle
   int v_steps=accuracy/10;
//--- déplace le premier point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
```

© 2000-2024, MetaQuotes Ltd.

```
      if(p1>1)
         p1-=1;
      //--- déplace le point
      if(!FiboExpansionPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   v_steps=accuracy/2;
//--- déplace le troisième point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p3>1)
         p3-=1;
      //--- déplace le point
      if(!FiboExpansionPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   v_steps=accuracy*4/5;
//--- déplace le second point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p2<accuracy-1)
         p2+=1;
      //--- déplace le point
      if(!FiboExpansionPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'objet du graphique
   FiboExpansionDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ELLIOTWAVE5

Vague d'Impulsion d'Elliot.

Note

Pour la "Vague d'Impulsion d'Elliott", il est possible d'activer/désactiver le mode de connexion des points par des lignes (propriété OBJPROP_DRAWLINES), mais aussi le niveau de positionnement de la vague (avec l'énumération ENUM_ELLIOT_WAVE_DEGREE).

Exemple

Le script suivant crée et déplace une vague d'impulsion d'Elliott sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un objet graphique \"Vague d'Impulsion d'Elli
#property description "Les coordonnées du point d'ancrage sont en pourcentage de la ta
#property description "de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string                InpName="ElliotWave5";   // nom de l'objet
input int                   InpDate1=10;             // Date du premier point, %
input int                   InpPrice1=90;            // Prix du premier point, %
input int                   InpDate2=20;             // Date du second point, %
input int                   InpPrice2=40;            // Prix du second point, %
input int                   InpDate3=30;             // Date du troisième point, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int                     InpPrice3=60;            // Prix du troisième point, %
input int                     InpDate4=40;             // Date du quatrième point, %
input int                     InpPrice4=10;            // Prix du quatrième point, %
input int                     InpDate5=60;             // Date du cinquième point, %
input int                     InpPrice5=40;            // Prix du cinquième, %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Niveau
input bool                    InpDrawLines=true;       // Affichage des lignes
input color                   InpColor=clrRed;         // Couleur des lignes
input ENUM_LINE_STYLE         InpStyle=STYLE_DASH;     // Style des lignes
input int                     InpWidth=2;              // Largeur des lignes
input bool                    InpBack=false;           // Objet en arrière plan
input bool                    InpSelection=true;       // mise en surbrillance pour le
input bool                    InpHidden=true;          // caché dans la liste des obje
input long                    InpZOrder=0;             // priorité pour le clic de sou
//+------------------------------------------------------------------------------------
//| Crée une "Vague d'Impulsion d'Elliott" aux coordonnées données
//+------------------------------------------------------------------------------------
bool ElliotWave5Create(const long                    chart_ID=0,              // ident
                       const string                  name="ElliotWave5",      // nom c
                       const int                     sub_window=0,            // indic
                       datetime                      time1=0,                 // heure
                       double                        price1=0,                // prix
                       datetime                      time2=0,                 // heure
                       double                        price2=0,                // prix
                       datetime                      time3=0,                 // heure
                       double                        price3=0,                // prix
                       datetime                      time4=0,                 // heure
                       double                        price4=0,                // prix
                       datetime                      time5=0,                 // heure
                       double                        price5=0,                // prix
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // degre
                       const bool                    draw_lines=true,         // Affic
                       const color                   clr=clrRed,              // coule
                       const ENUM_LINE_STYLE         style=STYLE_SOLID,       // Style
                       const int                     width=1,                 // Large
                       const bool                    back=false,              // en ar
                       const bool                    selection=true,          // mise
                       const bool                    hidden=true,             // caché
                       const long                    z_order=0)               // prior
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeElliotWave5EmptyPoints(time1,price1,time2,price2,time3,price3,time4,price4,ti
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- Crée une "Vague d'Impulsion d'Elliott" aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE5,sub_window,time1,price1,time2,price2
      price3,time4,price4,time5,price5))
     {
      Print(__FUNCTION__,
```

```
            " : échec de création de la \"Vague d'Impulsion d'Elliott\" ! Code d'erreu
      return(false);
     }
//--- définit le degré (taille de la vague)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- active (true) ou désactive (false) le mode d'affichage des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- définit la couleur de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du canal pour
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage de la Vague d'Impulsion d'Elliott
//+------------------------------------------------------------------------------------
bool ElliotWave5PointChange(const long   chart_ID=0,         // identifiant du graphiq
                            const string name="ElliotWave5", // nom de l'objet
                            const int    point_index=0,      // indice du point d'ancr
                            datetime     time=0,             // coordonnées du point o
                            double       price=0)            // coordonnées du point o
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
```

```
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime la Vague d'Impulsion d'Elliott
//+------------------------------------------------------------------------------------
bool ElliotWave5Delete(const long   chart_ID=0,         // identifiant du graphique
                       const string name="ElliotWave5") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de la suppression de la \"Vague d'Impulsion d'Elliott\" ! Code d
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de la Vague d'Impulsion d'Elliott
//| et définit les valeurs par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeElliotWave5EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3,
                                  datetime &time4,double &price4,
                                  datetime &time5,double &price5)
  {
//--- tableau pour récupérer l'heure d'ouverture des 10 dernières barres
   datetime temp[];
   ArrayResize(temp,10);
//--- récupère les données
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- récupère la valeur d'un point du graphique courant
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si l'heure du premier point n'est pas définie, il sera placé 9 barres à gauche d
   if(!time1)
      time1=temp[0];
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du second point n'est pas définie, il sera placé 7 barres à gauche de
   if(!time2)
      time2=temp[2];
```

© 2000-2024, MetaQuotes Ltd.

```
//--- si le prix du deuxième point n'est pas défini, le déplace 300 points plus bas qu
   if(!price2)
      price2=price1-300*point;
//--- si l'heure du troisième point n'est pas définie, il sera placé 5 barres à gauche
   if(!time3)
      time3=temp[4];
//--- si le prix du troisième point n'est pas défini, le déplace de 250 points plus ba
   if(!price3)
      price3=price1-250*point;
//--- si l'heure du quatrième point n'est pas définie, il sera placé 3 barres à gauche
   if(!time4)
      time4=temp[6];
//--- si le prix du quatrième point n'est pas défini, le déplace de 550 points plus ba
   if(!price4)
      price4=price1-550*point;
//--- si l'heure du cinquième point n'est pas définie, elle sera définie sur la barre
   if(!time5)
      time5=temp[9];
//--- si le prix du cinquième point n'est pas défini, le déplace de 450 points plus ba
   if(!price5)
      price5=price1-450*point;
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100 ||
      InpDate4<0 || InpDate4>100 || InpPrice4<0 || InpPrice4>100 ||
      InpDate5<0 || InpDate5>100 || InpPrice5<0 || InpPrice5>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage de l'objet
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
```

© 2000-2024, MetaQuotes Ltd.

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner la Vague d'Impulsion d'Elliott
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int d4=InpDate4*(bars-1)/100;
   int d5=InpDate5*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
   int p4=InpPrice4*(accuracy-1)/100;
   int p5=InpPrice5*(accuracy-1)/100;
//--- Crée la Vague d'Impulsion d'Elliott
   if(!ElliotWave5Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],p
      date[d4],price[p4],date[d5],price[p5],InpDegree,InpDrawLines,InpColor,InpStyle,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage
//--- compteur de boucle
   int v_steps=accuracy/5;
//--- déplace le cinquième point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p5<accuracy-1)
         p5+=1;
      //--- déplace le point
      if(!ElliotWave5PointChange(0,InpName,4,date[d5],price[p5]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
```

© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   v_steps=accuracy/5;
//--- déplace le deuxième et le troisième points d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise les valeurs suivantes
      if(p2<accuracy-1)
         p2+=1;
      if(p3>1)
         p3-=1;
      //--- décale les points
      if(!ElliotWave5PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      if(!ElliotWave5PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   v_steps=accuracy*4/5;
//--- déplace le premier et le quatrième points d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise les valeurs suivantes
      if(p1>1)
         p1-=1;
      if(p4<accuracy-1)
         p4+=1;
      //--- décale les points
      if(!ElliotWave5PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave5PointChange(0,InpName,3,date[d4],price[p4]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'objet du graphique
   ElliotWave5Delete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ELLIOTWAVE3

Vague de Correction d'Elliott.

Note

Pour la "Vague de Correction d'Elliott", il est possible d'activer/désactiver le mode de connexion des points par des lignes (propriété OBJPROP_DRAWLINES), mais aussi le niveau de positionnement de la vague (avec l'énumération ENUM_ELLIOT_WAVE_DEGREE).

Exemple

Le script suivant crée et déplace une vague de correction d'Elliott sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un objet graphique \"Vague de Correction d'El
#property description "Les coordonnées du point d'ancrage sont définies en pourcentage
#property description "la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string                InpName="ElliotWave3";   // nom de l'objet
input int                   InpDate1=10;             // Date du premier point, %
input int                   InpPrice1=90;            // Prix du premier point, %
input int                   InpDate2=30;             // Date du second point, %
input int                   InpPrice2=10;            // Prix du second point, %
input int                   InpDate3=50;             // Date du troisième point, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int                     InpPrice3=40;            // Prix du troisième point, %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Niveau
input bool                    InpDrawLines=true;       // Affichage des lignes
input color                   InpColor=clrRed;         // Couleur des lignes
input ENUM_LINE_STYLE         InpStyle=STYLE_DASH;     // Style des lignes
input int                     InpWidth=2;              // Largeur des lignes
input bool                    InpBack=false;           // Objet en arrière plan
input bool                    InpSelection=true;       // mise en surbrillance pour le
input bool                    InpHidden=true;          // caché dans la liste des obje
input long                    InpZOrder=0;             // priorité pour le clic de sou
//+------------------------------------------------------------------------------------
//| Crée une "Vague de Correction d'Elliott" aux coordonnées données
//+------------------------------------------------------------------------------------
bool ElliotWave3Create(const long                    chart_ID=0,              // ident
                       const string                  name="ElliotWave3",      // nom d
                       const int                     sub_window=0,            // indic
                       datetime                      time1=0,                 // heure
                       double                        price1=0,                // prix
                       datetime                      time2=0,                 // heure
                       double                        price2=0,                // prix
                       datetime                      time3=0,                 // heure
                       double                        price3=0,                // prix
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // degré
                       const bool                    draw_lines=true,         // Affic
                       const color                   clr=clrRed,              // coule
                       const ENUM_LINE_STYLE         style=STYLE_SOLID,       // Style
                       const int                     width=1,                 // Large
                       const bool                    back=false,              // en ar
                       const bool                    selection=true,          // mise
                       const bool                    hidden=true,             // caché
                       const long                    z_order=0)               // prior
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeElliotWave3EmptyPoints(time1,price1,time2,price2,time3,price3);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- Crée une "Vague de Correction d'Elliott" aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE3,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            " : échec de création de la \"Vague de Correction d'Elliott\" ! Code d'err
      return(false);
     }
//--- définit le degré (taille de la vague)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- active (true) ou désactive (false) le mode d'affichage des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- définit la couleur de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du canal pour
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage de la Vague de Correction d'Elliott
//+------------------------------------------------------------------------------------
bool ElliotWave3PointChange(const long   chart_ID=0,         // identifiant du graphiq
                            const string name="ElliotWave3", // nom de l'objet
                            const int    point_index=0,      // indice du point d'ancr
                            datetime     time=0,             // coordonnées du point o
                            double       price=0)            // coordonnées du point o
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime la Vague de Correction d'Elliott
//+------------------------------------------------------------------------------------
bool ElliotWave3Delete(const long   chart_ID=0,         // identifiant du graphique
```

© 2000-2024, MetaQuotes Ltd.

```
                        const string name="ElliotWave3") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de la suppression de la \"Vague de Correction d'Elliott\" ! Code
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de la Vague de Correction d'Elliott
//| et définit les valeurs par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeElliotWave3EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3)
  {
//--- tableau pour récupérer l'heure d'ouverture des 10 dernières barres
   datetime temp[];
   ArrayResize(temp,10);
//--- récupère les données
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- récupère la valeur d'un point du graphique courant
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si l'heure du premier point n'est pas définie, il sera placé 9 barres à gauche d
   if(!time1)
      time1=temp[0];
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du second point n'est pas définie, il sera placé 5 barres à gauche de
   if(!time2)
      time2=temp[4];
//--- si le prix du deuxième point n'est pas défini, le déplace 300 points plus bas qu
   if(!price2)
      price2=price1-300*point;
//--- si l'heure du troisième point n'est pas définie, il sera placé 1 barre à gauche
   if(!time3)
      time3=temp[8];
//--- si le prix du troisième point n'est pas défini, le déplace de 200 points plus ba
   if(!price3)
      price3=price1-200*point;
  }
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage de l'objet
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner la Vague de Correction d'Elliott
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- Crée la Vague de Correction d'Elliott
   if(!ElliotWave3Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      InpDegree,InpDrawLines,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidder
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage
//--- compteur de boucle
   int v_steps=accuracy/5;
//--- déplace le troisième point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p3<accuracy-1)
         p3+=1;
      //--- déplace le point
      if(!ElliotWave3PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   v_steps=accuracy*4/5;
//--- déplace le premier et le deuxième point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise les valeurs suivantes
      if(p1>1)
         p1-=1;
      if(p2<accuracy-1)
         p2+=1;
      //--- décale les points
      if(!ElliotWave3PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave3PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'objet du graphique
```

© 2000-2024, MetaQuotes Ltd.

```
   ElliotWave3Delete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_RECTANGLE

Rectangle.

Note

Pour le rectangle, le mode de remplissage avec une couleur peut être défini avec la propriété OBJPROP_FILL.

Exemple

Le script suivant crée et déplace un rectangle sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée un rectangle sur le graphique."
#property description "Les coordonnées du point d'ancrage sont définies en"
#property description "pourcentage de la taille de la fenêtre du graphique"
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="Rectangle"; // Nom du rectangle
input int             InpDate1=40;         // Date du premier point, %
input int             InpPrice1=40;        // Prix du premier point, %
input int             InpDate2=60;         // Date du second point, %
input int             InpPrice2=60;        // Prix du second point, %
input color           InpColor=clrRed;     // Couleur du rectangle
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Style des lignes du rectangle
```

© 2000-2024, MetaQuotes Ltd.

```
input int              InpWidth=2;          // Largeur des lignes du rectangle
input bool             InpFill=true;        // Remplissage du rectangle avec une couleu
input bool             InpBack=false;       // Rectangle en arrière plan
input bool             InpSelection=true;   // Mise en surbrillance pour déplacer
input bool             InpHidden=true;      // Caché dans la liste des objets
input long             InpZOrder=0;         // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée un rectangle aux coordonnées données
//+------------------------------------------------------------------------------------
bool RectangleCreate(const long            chart_ID=0,        // identifiant du graphi
                     const string          name="Rectangle",  // nom du rectangle
                     const int             sub_window=0,      // indice de sous-fenêtr
                     datetime              time1=0,           // heure du premier poir
                     double                price1=0,          // prix du premier point
                     datetime              time2=0,           // heure du second point
                     double                price2=0,          // prix du second point
                     const color           clr=clrRed,        // couleur du rectangle
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // style de lignes du re
                     const int             width=1,           // largeur des lignes du
                     const bool            fill=false,        // remplissage du rectar
                     const bool            back=false,        // en arrière-plan
                     const bool            selection=true,    // mise en surbrillance
                     const bool            hidden=true,       // caché dans la liste d
                     const long            z_order=0)         // priorité du clic de s
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeRectangleEmptyPoints(time1,price1,time2,price2);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée un rectangle aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            " : impossible de créer un rectangle ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- définit la couleur du rectangle
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style des lignes du rectangle
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes du rectangle
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- active (true) ou désactive (false) le mode de remplissage du rectangle
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du rectangle
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
```

© 2000-2024, MetaQuotes Ltd.

```
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage du rectangle
//+------------------------------------------------------------------------------------
bool RectanglePointChange(const long   chart_ID=0,       // identifiant du graphique
                          const string name="Rectangle", // nom du rectangle
                          const int    point_index=0,    // indice du point d'ancrage
                          datetime     time=0,           // coordonnée horaire du poin
                          double       price=0)          // coordonnée prix du point o
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le rectangle
//+------------------------------------------------------------------------------------
bool RectangleDelete(const long   chart_ID=0,       // identifiant du graphique
                     const string name="Rectangle") // nom du rectangle
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime le rectangle
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": impossible de supprimer le rectangle ! Code d'erreur = ",GetLastError()
```

© 2000-2024, MetaQuotes Ltd.

```
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage du rectangle et définit les valeurs
//| par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeRectangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- si l'heure du premier point n'est pas définie, elle sera définie sur la barre co
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du second point n'est pas réglée, elle se trouve à 9 barres de la sec
   if(!time2)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- Positionne le deuxième point à 9 barres à gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas défini, le déplace 300 points plus bas qu
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage du rectangle
```

© 2000-2024, MetaQuotes Ltd.

```
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner le rectangle
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crée un rectangle
   if(!RectangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplace les points d'ancrage du rectangle
//--- compteur de boucle
   int h_steps=bars/2;
//--- déplace les points d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise les valeurs suivantes
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- décale les points
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
```

© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   int v_steps=accuracy/2;
//--- déplace les points d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise les valeurs suivantes
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- décale les points
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime le rectangle du graphique
   RectangleDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_TRIANGLE

Triangle.

Note

Pour le triangle, le mode de remplissage avec une couleur peut être défini avec la propriété OBJPROP_FILL.

Exemple

Le script suivant crée et déplace un triangle sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée un triangle sur le graphique."
#property description "Les coordonnées du point d'ancrage sont définies en"
#property description "pourcentage de la taille de la fenêtre du graphique"
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="Triangle";        // Nom du triangle
input int             InpDate1=25;               // Date du premier point, %
input int             InpPrice1=50;              // Prix du premier point, %
input int             InpDate2=70;               // Date du second point, %
input int             InpPrice2=70;              // Prix du second point, %
input int             InpDate3=65;               // Date du troisième point, %
input int             InpPrice3=20;              // Prix du troisième point, %
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // Couleur du triangle
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style de lignes du triangle
input int             InpWidth=2;                // Largeur de lignes du triangle
input bool            InpFill=false;             // Remplissage du triangle avec une c
input bool            InpBack=false;             // Triangle en arrière-plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplacer
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée un triangle aux coordonnées données
//+------------------------------------------------------------------------------------
bool TriangleCreate(const long            chart_ID=0,        // identifiant du graphic
                    const string          name="Triangle",   // nom du triangle
                    const int             sub_window=0,      // indice de sous-fenêtre
                    datetime              time1=0,           // heure du premier point
                    double                price1=0,          // prix du premier point
                    datetime              time2=0,           // heure du deuxième poir
                    double                price2=0,          // prix du deuxième point
                    datetime              time3=0,           // heure du troisième poi
                    double                price3=0,          // prix du troisième poir
                    const color           clr=clrRed,        // couleur du triangle
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // style de lignes du tri
                    const int             width=1,           // largeur de lignes du t
                    const bool            fill=false,        // remplissage du triangl
                    const bool            back=false,        // en arrière-plan
                    const bool            selection=true,    // mise en surbrillance p
                    const bool            hidden=true,       // caché dans la liste de
                    const long            z_order=0)         // priorité du clic de sc
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeTriangleEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée un triangle par les coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_TRIANGLE,sub_window,time1,price1,time2,price2,ti
     {
      Print(__FUNCTION__,
            " : impossible de créer un triangle ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- définit la couleur du triangle
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style des lignes du triangle
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes du triangle
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- active (true) ou désactive (false) le mode de remplissage du triangle
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affiche en premier plan (false) ou en arrière-plan (true)
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de mise en surbrillance du triangle p
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage du triangle
//+------------------------------------------------------------------------------------
bool TrianglePointChange(const long   chart_ID=0,      // identifiant du graphique
                         const string name="Triangle", // nom du triangle
                         const int    point_index=0,   // indice du point d'ancrage
                         datetime     time=0,          // coordonnées du point d'ancra
                         double       price=0)         // coordonnées du point d'ancra
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le triangle
//+------------------------------------------------------------------------------------
bool TriangleDelete(const long   chart_ID=0,      // identifiant du graphique
                    const string name="Triangle") // nom du triangle
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime le triangle
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : impossible de supprimer l'ellipse ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage du triangle et définit les valeurs
//| par défaut des points vides
//+------------------------------------------------------------------------------------
void ChangeTriangleEmptyPoints(datetime &time1,double &price1,
                               datetime &time2,double &price2,
                               datetime &time3,double &price3)
  {
//--- si l'heure du premier point n'est pas définie, elle sera définie sur la barre co
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du second point n'est pas réglée, elle se trouve à 9 barres de la sec
   if(!time2)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- Positionne le deuxième point à 9 barres à gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas défini, le déplace 300 points plus bas qu
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si l'heure du troisième point n'est pas définie, il sera mis sur la date du deux
   if(!time3)
      time3=time2;
//--- si le prix du troisième point n'est pas défini, il est égal à celui du premier p
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
```

```
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage du triangle
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner le triangle
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crée un triangle
   if(!TriangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage du triangle
```

© 2000-2024, MetaQuotes Ltd.

```
//--- compteur de boucle
   int v_steps=accuracy*3/10;
//--- déplace le premier point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p1>1)
         p1-=1;
      //--- déplace le point
      if(!TrianglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   int h_steps=bars*9/20-1;
//--- déplace le second point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d2>1)
         d2-=1;
      //--- déplace le point
      if(!TrianglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   v_steps=accuracy/4;
//--- déplace le troisième point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p3<accuracy-1)
         p3+=1;
      //--- déplace le point
```

© 2000-2024, MetaQuotes Ltd.

```
      if(!TrianglePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime le triangle du graphique
   TriangleDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ELLIPSE

Ellipse.

Note

Pour l'ellipse, le mode de remplissage avec une couleur peut être défini avec la propriété OBJPROP_FILL.

Exemple

Le script suivant crée et déplace une ellipse sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée une ellipse sur le graphique."
#property description "Les coordonnées du point d'ancrage sont définies"
#property description "en pourcentage de la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="Ellipse";         // Nom de l'ellipse
input int             InpDate1=30;               // Date du premier point, %
input int             InpPrice1=20;              // Prix du premier point, %
input int             InpDate2=70;               // Date du second point, %
input int             InpPrice2=80;              // Prix du second point, %
input int             InpDate3=50;               // Date du troisième point, %
input int             InpPrice3=60;              // Prix du troisième point, %
input color           InpColor=clrRed;           // Couleur de l'ellipse
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Style des lignes de l'ellipse
input int             InpWidth=2;                // Largeur des lignes de l'ellipse
input bool            InpFill=false;             // Remplissage de l'ellipse avec une
input bool            InpBack=false;             // Ellipse en arrière plan
input bool            InpSelection=true;         // Mise en surbrillance pour déplacer
input bool            InpHidden=true;            // Caché dans la liste des objets
input long            InpZOrder=0;               // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée une ellipse aux coordonnées données
//+------------------------------------------------------------------------------------
bool EllipseCreate(const long            chart_ID=0,        // identifiant du graphiqu
                   const string          name="Ellipse",    // Nom de l'ellipse
                   const int             sub_window=0,      // indice de sous-fenêtre
                   datetime              time1=0,           // heure du premier point
                   double                price1=0,          // prix du premier point
                   datetime              time2=0,           // heure du deuxième point
                   double                price2=0,          // prix du deuxième point
                   datetime              time3=0,           // heure du troisième poir
                   double                price3=0,          // prix du troisième point
                   const color           clr=clrRed,        // couleur de l'ellipse
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // style des lignes de l'e
                   const int             width=1,           // largeur des lignes de l
                   const bool            fill=false,        // remplissage de l'ellips
                   const bool            back=false,        // en arrière plan
                   const bool            selection=true,    // mise en surbrillance po
                   const bool            hidden=true,       // caché dans la liste des
                   const long            z_order=0)         // priorité pour le clic c
  {
//--- définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeEllipseEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée une ellipse aux coordonnées données
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIPSE,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            " : échec de création d'une ellipse ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- définit la couleur de l'ellipse
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style des lignes de l'ellipse
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur des lignes de l'ellipse
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- active (true) ou désactive (false) le mode de remplissage de l'ellipse
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- active (true) ou désactive (false) le mode de mise en surbrillance de l'ellipse
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage d l'ellipse
//+------------------------------------------------------------------------------------
bool EllipsePointChange(const long   chart_ID=0,     // identifiant du graphique
                        const string name="Ellipse", // nom de l'ellipse
                        const int    point_index=0,  // indice du point d'ancrage
                        datetime     time=0,         // coordonnées du point d'ancrage
                        double       price=0)        // coordonnées du point d'ancrage
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime l'ellipse
//+------------------------------------------------------------------------------------
bool EllipseDelete(const long   chart_ID=0,     // identifiant du graphique
                   const string name="Ellipse") // nom de l'ellipse
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'ellipse
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             " : échec de suppression d'une ellipse ! Code d'erreur = ",GetLastError())
       return(false);
      }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs des points d'ancrage de l'ellipse et définit les valeurs par d
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeEllipseEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2,
                              datetime &time3,double &price3)
  {
//--- si l'heure du premier point n'est pas définie, elle sera définie sur la barre co
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas défini, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si l'heure du second point n'est pas réglée, elle se trouve à 9 barres de la sec
   if(!time2)
     {
      //--- tableau pour recevoir l'heure d'ouverture des 10 dernières barres
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- Positionne le deuxième point à 9 barres à gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas défini, le déplace 300 points plus bas qu
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si l'heure du troisième point n'est pas définie, il sera mis sur la date du deux
   if(!time3)
      time3=time2;
//--- si le prix du troisième point n'est pas défini, il est égal à celui du premier p
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
```

© 2000-2024, MetaQuotes Ltd.

```
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage de l'ellipse
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner l'ellipse
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crée une ellipse
   if(!EllipseCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant les points d'ancrage de l'ellipse
//--- compteur de boucle
```

© 2000-2024, MetaQuotes Ltd.

```
   int v_steps=accuracy/5;
//--- déplace le premier et le deuxième point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise les valeurs suivantes
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- décale les points
      if(!EllipsePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!EllipsePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   int h_steps=bars/5;
//--- déplace le troisième point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d3>1)
         d3-=1;
      //--- déplace le point
      if(!EllipsePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'ellipse du graphique
   EllipseDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_THUMB_UP

Signe Pouce vers le Haut.

Note

La position du point d'ancrage par rapport au signe peut être sélectionnée dans l'énumération ENUM_ARROW_ANCHOR.

Les gros signes (plus de 5) peuvent être créés simplement en définissant la valeur de la propriété OBJPROP_WIDTH correspondante lors de l'édition du code dans MetaEditor.

Exemple

Le script suivant crée et déplace le signe Pouce vers le Haut sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine le signe \"Pouce vers le Haut\"."
#property description "La coordonnée du point d'ancrage est définie en pourcentage de'
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="ThumbUp";     // nom du signe
input int               InpDate=75;            // Date du point d'ancrage en %
input int               InpPrice=25;           // Prix du point d'ancrage en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP;  // type d'ancrage
input color             InpColor=clrRed;       // Couleur du signe
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Style de la ligne de bordure
input int               InpWidth=5;           // Taille du signe
input bool              InpBack=false;        // Signe en arrière plan
input bool              InpSelection=true;    // Mise en surbrillance pour déplacer
input bool              InpHidden=true;       // Caché dans la liste des objets
input long              InpZOrder=0;          // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée le signe Pouce vers le Haut
//+------------------------------------------------------------------------------------
bool ArrowThumbUpCreate(const long              chart_ID=0,           // identifiant d
                        const string            name="ThumbUp",       // nom du signe
                        const int               sub_window=0,         // indice de sou
                        datetime                time=0,               // heure du poin
                        double                  price=0,              // prix du point
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // type d'ancrag
                        const color             clr=clrRed,           // Couleur du si
                        const ENUM_LINE_STYLE   style=STYLE_SOLID,    // Style de la l
                        const int               width=3,              // Taille du sig
                        const bool              back=false,           // en arrière-pl
                        const bool              selection=true,       // mise en surbr
                        const bool              hidden=true,          // caché dans la
                        const long              z_order=0)            // priorité du d
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer le signe \"Pouce vers le Haut\" ! Code d'erreur
      return(false);
     }
//--- définit le type d'ancrage
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- définit la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (vrai) ou désactive (faux) le mode de déplacement du signal par la souris
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
bool ArrowThumbUpMove(const long   chart_ID=0,      // identifiant du graphique
                      const string name="ThumbUp", // nom de l'objet
                      datetime     time=0,         // coordonnées du point d'ancrage t
                      double       price=0)        // coordonnées du point d'ancrage p
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie le type d'ancrage du signe Pouce vers le Haut
//+------------------------------------------------------------------------------------
bool ArrowThumbUpAnchorChange(const long              chart_ID=0,        // identifiar
                              const string            name="ThumbUp",    // nom de l'o
                              const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // type d'anc
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le type d'ancrage
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le type d'ancrage ! Code d'erreur = ",GetLastEr
      return(false);
     }
```

```
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le signe Pouce vers le Haut
//+------------------------------------------------------------------------------------
bool ArrowThumbUpDelete(const long   chart_ID=0,     // identifiant du graphique
                        const string name="ThumbUp") // nom du signe
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- efface le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à supprimer le signe \"Pouce vers le Haut\" ! Code d'er
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées du point d'ancrage du signe
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points en dessinant le signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crée le signe Pouce vers le Haut sur le graphique
   if(!ArrowThumbUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplacez le point d'ancrage et changez sa position par rapport au si
//--- compteur de boucle
   int h_steps=bars/4;
//--- déplace le point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d>1)
         d-=1;
      //--- déplace le point
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   int v_steps=accuracy/4;
//--- déplace le point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p<accuracy-1)
         p+=1;
      //--- déplace le point
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- modifie l'emplacement du point d'ancrage par rapport au signal
   ArrowThumbUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redessine le graphique
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//--- efface le signe du graphique
   ArrowThumbUpDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_THUMB_DOWN

Signe du symbole Pouce en Bas

Note

La position du point d'ancrage par rapport au signe peut être sélectionnée dans l'énumération ENUM_ARROW_ANCHOR.

Les gros signes (plus de 5) peuvent être créés simplement en définissant la valeur de la propriété OBJPROP_WIDTH correspondante lors de l'édition du code dans MetaEditor.

Exemple

Le script suivant crée et déplace le signe Pouce en bas sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine \"Pouce vers le Bas\"."
#property description "La coordonnée du point d'ancrage est définie en pourcentage de'
#property description "la taille de la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="ThumbDown";      // nom du signe
input int               InpDate=25;               // Date du point d'ancrage en %
input int               InpPrice=75;              // Prix du point d'ancrage en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // type d'ancrage
```

© 2000-2024, MetaQuotes Ltd.

```
input color             InpColor=clrRed;         // Couleur du signe
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Style de la ligne de bordure
input int               InpWidth=5;              // Taille du signe
input bool              InpBack=false;           // Signe en arrière plan
input bool              InpSelection=true;       // Mise en surbrillance pour déplacer
input bool              InpHidden=true;          // Caché dans la liste des objets
input long              InpZOrder=0;             // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée le signe Pouce en Bas
//+------------------------------------------------------------------------------------
bool ArrowThumbDownCreate(const long              chart_ID=0,           // identifiant
                          const string            name="ThumbDown",     // nom du sign
                          const int               sub_window=0,         // indice de s
                          datetime                time=0,               // heure du po
                          double                  price=0,              // prix du poi
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // type d'ancr
                          const color             clr=clrRed,           // Couleur du
                          const ENUM_LINE_STYLE   style=STYLE_SOLID,    // Style de la
                          const int               width=3,              // Taille du s
                          const bool              back=false,           // en arrière
                          const bool              selection=true,       // mise en sur
                          const bool              hidden=true,          // caché dans
                          const long              z_order=0)            // priorité du
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer le signe \" Pouce en Bas\" ! Code d'erreur = ",
      return(false);
     }
//--- définit le type d'ancrage
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- définit la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (vrai) ou désactive (faux) le mode de déplacement du signal par la souris
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des c
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
bool ArrowThumbDownMove(const long   chart_ID=0,        // identifiant du graphique
                        const string name="ThumbDown", // nom de l'objet
                        datetime     time=0,            // coordonnées du point d'ancra
                        double       price=0)           // coordonnées du point d'ancra
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie le type d'ancrage du signe Pouce en Bas
//+------------------------------------------------------------------------------------
bool ArrowThumbDownAnchorChange(const long              chart_ID=0,        // identifi
                                const string            name="ThumbDown",  // nom de l
                                const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // type d'a
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le type d'ancrage
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le type d'ancrage ! Code d'erreur = ",GetLastEr
      return(false);
```

```
      }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le signe Pouce en Bas
//+------------------------------------------------------------------------------------
bool ArrowThumbDownDelete(const long   chart_ID=0,       // identifiant du graphique
                          const string name="ThumbDown") // nom du signe
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- efface le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": n'a pas réussi à supprimer le signe \"Pouce en Bas\" sign! Code d'erreu
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
```

© 2000-2024, MetaQuotes Ltd.

```
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées du point d'ancrage du signe
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points en dessinant le signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crée le signe Pouce en Bas sur le graphique
   if(!ArrowThumbDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplacez le point d'ancrage et changez sa position par rapport au si
//--- compteur de boucle
   int h_steps=bars/4;
//--- déplace le point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- déplace le point
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
```

© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- compteur de boucle
   int v_steps=accuracy/4;
//--- déplace le point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p>1)
         p-=1;
      //--- déplace le point
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- modifie l'emplacement du point d'ancrage par rapport au signal
   ArrowThumbDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- redessine le graphique
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//--- efface le signe du graphique
   ArrowThumbDownDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_UP

Signe Flèche vers le Haut.

Note

La position du point d'ancrage par rapport au signe peut être sélectionnée dans l'énumération ENUM_ARROW_ANCHOR.

Les gros signes (plus de 5) peuvent être créés simplement en définissant la valeur de la propriété OBJPROP_WIDTH correspondante lors de l'édition du code dans MetaEditor.

Exemple

Le script suivant crée et déplace le signe Flèche vers le Haut sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine le signe \"Flèche vers le Haut\"."
#property description "La coordonnée du point d'ancrage est définie en"
#property description "pourcentage de la taille de la fenêtre du graphique"
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="ArrowUp";     // nom du signe
input int               InpDate=25;            // Date du point d'ancrage en %
input int               InpPrice=25;           // Prix du point d'ancrage en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP;  // type d'ancrage
input color             InpColor=clrRed;       // Couleur du signe
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Style de la ligne de bordure
input int               InpWidth=5;           // Taille du signe
input bool              InpBack=false;        // Signe en arrière plan
input bool              InpSelection=false;   // Mise en surbrillance pour déplacer
input bool              InpHidden=true;       // Caché dans la liste des objets
input long              InpZOrder=0;          // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée le signe Flèche vers le Haut
//+------------------------------------------------------------------------------------
bool ArrowUpCreate(const long              chart_ID=0,           // identifiant du gra
                   const string            name="ArrowUp",       // nom du signe
                   const int               sub_window=0,         // indice de sous-fen
                   datetime                time=0,               // heure du point d'a
                   double                  price=0,              // prix du point d'ar
                   const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // type d'ancrage
                   const color             clr=clrRed,           // Couleur du signe
                   const ENUM_LINE_STYLE   style=STYLE_SOLID,    // Style de la ligne
                   const int               width=3,              // Taille du signe
                   const bool              back=false,           // en arrière-plan
                   const bool              selection=true,       // mise en surbrillan
                   const bool              hidden=true,          // caché dans la list
                   const long              z_order=0)            // priorité du clic d
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer le signe \"Flèche vers le Haut\" ! Code d'erreu
      return(false);
     }
//--- définit le type d'ancrage
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- définit la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (vrai) ou désactive (faux) le mode de déplacement du signal par la souris
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
bool ArrowUpMove(const long   chart_ID=0,     // identifiant du graphique
                 const string name="ArrowUp", // nom de l'objet
                 datetime     time=0,         // coordonnées du point d'ancrage tempor
                 double       price=0)        // coordonnées du point d'ancrage prix
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie le type d'ancrage du signe Flèche vers le Haut
//+------------------------------------------------------------------------------------
bool ArrowUpAnchorChange(const long              chart_ID=0,        // identifiant du
                         const string            name="ArrowUp",    // nom de l'objet
                         const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // type d'ancrage
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie l'emplacement du point d'ancrage
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le type d'ancrage ! Code d'erreur = ",GetLastEr
      return(false);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le signe Flèche vers le Haut
//+------------------------------------------------------------------------------------
bool ArrowUpDelete(const long   chart_ID=0,     // identifiant du graphique
                   const string name="ArrowUp") // nom du signe
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- efface le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à supprimer le signe \"Flèche vers le Haut\" ! Code d'e
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées du point d'ancrage du signe
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points en dessinant le signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crée le signe Arrow Up sur le graphique
   if(!ArrowUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplacez le point d'ancrage et changez sa position par rapport au si
//--- compteur de boucle
   int v_steps=accuracy/2;
//--- déplace le point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p<accuracy-1)
         p+=1;
      //--- déplace le point
      if(!ArrowUpMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- modifie l'emplacement du point d'ancrage par rapport au signal
   ArrowUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redessine le graphique
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//--- efface le signe du graphique
   ArrowUpDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_DOWN

Signe Flèche vers le Bas.

Note

La position du point d'ancrage par rapport au signe peut être sélectionnée dans l'énumération ENUM_ARROW_ANCHOR.

Les gros signes (plus de 5) peuvent être créés simplement en définissant la valeur de la propriété OBJPROP_WIDTH correspondante lors de l'édition du code dans MetaEditor.

Exemple

Le script suivant crée et déplace le signe Flèche vers le Bas sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un signe \"Flèche vers le Bas\" signe."
#property description "La coordonnée du point d'ancrage est définie en"
#property description "pourcentage de la taille de la fenêtre du graphique"
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="ArrowDown";      // nom du signe
input int               InpDate=75;               // Date du point d'ancrage en %
input int               InpPrice=75;              // Prix du point d'ancrage en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM;  // type d'ancrage
input color             InpColor=clrRed;          // Couleur du signe
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;       // Style de la ligne de bordure
```

© 2000-2024, MetaQuotes Ltd.

```
input int                InpWidth=5;              // Taille du signe
input bool               InpBack=false;           // Signe en arrière plan
input bool               InpSelection=false;      // Mise en surbrillance pour déplacer
input bool               InpHidden=true;          // Caché dans la liste des objets
input long               InpZOrder=0;             // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée le signe Flèche vers le Bas
//+------------------------------------------------------------------------------------
bool ArrowDownCreate(const long              chart_ID=0,           // identifiant du g
                     const string            name="ArrowDown",     // nom du signe
                     const int               sub_window=0,         // indice de sous-f
                     datetime                time=0,               // heure du point d
                     double                  price=0,              // prix du point d'
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // type d'ancrage
                     const color             clr=clrRed,           // Couleur du signe
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // Style de la ligr
                     const int               width=3,              // Taille du signe
                     const bool              back=false,           // en arrière-plan
                     const bool              selection=true,       // mise en surbrill
                     const bool              hidden=true,          // caché dans la li
                     const long              z_order=0)            // priorité du clic
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer le signal \"Flèche vers le Bas\"! Code d'erreur
      return(false);
     }
//--- type d'ancrage
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- définit la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (vrai) ou désactive (faux) le mode de déplacement du signal par la souris
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des c
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
bool ArrowDownMove(const long   chart_ID=0,       // identifiant du graphique
                   const string name="ArrowDown", // nom de l'objet
                   datetime     time=0,           // coordonnées du point d'ancrage te
                   double       price=0)          // coordonnées du point d'ancrage pr
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie le type d'ancrage du signe Flèche vers le Haut
//+------------------------------------------------------------------------------------
bool ArrowDownAnchorChange(const long              chart_ID=0,        // identifiant d
                           const string            name="ArrowDown",  // nom de l'obje
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // type d'ancrag
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie l'emplacement du point d'ancrage
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le type d'ancrage ! Code d'erreur = ",GetLastEr
      return(false);
     }
//--- exécution réussie
```

© 2000-2024, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le signe Flèche vers le Bas
//+------------------------------------------------------------------------------------
bool ArrowDownDelete(const long   chart_ID=0,       // identifiant du graphique
                     const string name="ArrowDown") // nom du signe
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- efface le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à supprimer le signe \"Flèche vers le Bas\"! Code d'er
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
```

© 2000-2024, MetaQuotes Ltd.

```
//--- pour le réglage et la modification des coordonnées du point d'ancrage du signe
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points en dessinant le signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crée le signe Arrow Down sur le grahique
   if(!ArrowDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplacez le point d'ancrage et changez sa position par rapport au si
//--- compteur de boucle
   int v_steps=accuracy/2;
//--- déplace le point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p>1)
         p-=1;
      //--- déplace le point
      if(!ArrowDownMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
```

© 2000-2024, MetaQuotes Ltd.

```
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- modifie l'emplacement du point d'ancrage par rapport au signal
   ArrowDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- redessine le graphique
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//--- efface le signe du graphique
   ArrowDownDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_STOP

Signe Stop.

Note

La position du point d'ancrage par rapport au signe peut être sélectionnée dans l'énumération ENUM_ARROW_ANCHOR.

Les gros signes (plus de 5) peuvent être créés simplement en définissant la valeur de la propriété OBJPROP_WIDTH correspondante lors de l'édition du code dans MetaEditor.

Exemple

Le script suivant crée et déplace le signe Stop sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine le signe \"Stop\"."
#property description "La coordonnée du point d'ancrage est définie en"
#property description "pourcentage de la taille de la fenêtre du graphique"
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="ArrowStop";      // nom du signe
input int               InpDate=10;               // Date du point d'ancrage en %
input int               InpPrice=50;              // Prix du point d'ancrage en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM;  // type d'ancrage
input color             InpColor=clrRed;          // Couleur du signe
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Style de la ligne de bordure
input int               InpWidth=5;              // Taille du signe
input bool              InpBack=false;           // Signe en arrière plan
input bool              InpSelection=false;      // Mise en surbrillance pour déplacer
input bool              InpHidden=true;          // Caché dans la liste des objets
input long              InpZOrder=0;             // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée un signe Stop
//+------------------------------------------------------------------------------------
bool ArrowStopCreate(const long              chart_ID=0,           // identifiant du g
                     const string            name="ArrowStop",     // nom du signe
                     const int               sub_window=0,         // indice de sous-f
                     datetime                time=0,               // heure du point d
                     double                  price=0,              // prix du point d'
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // type d'ancrage
                     const color             clr=clrRed,           // Couleur du signe
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // Style de la ligr
                     const int               width=3,              // Taille du signe
                     const bool              back=false,           // en arrière-plan
                     const bool              selection=true,       // mise en surbrill
                     const bool              hidden=true,          // caché dans la li
                     const long              z_order=0)            // priorité du clic
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_STOP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer le signe \"Stop\" Code d'erreur = ",GetLastErro
      return(false);
     }
//--- définit le type d'ancrage
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- définit la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (vrai) ou désactive (faux) le mode de déplacement du signal par la souris
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
bool ArrowStopMove(const long   chart_ID=0,       // identifiant du graphique
                   const string name="ArrowStop", // nom de l'objet
                   datetime     time=0,           // coordonnées du point d'ancrage te
                   double       price=0)          // coordonnées du point d'ancrage pr
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie le type d'ancrage du signe Stop
//+------------------------------------------------------------------------------------
bool ArrowStopAnchorChange(const long              chart_ID=0,        // identifiant d
                           const string            name="ArrowStop",  // nom de l'obje
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // position du p
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le type d'ancrage
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le type d'ancrage ! Code d'erreur = ",GetLastEr
      return(false);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le signe Stop
//+------------------------------------------------------------------------------------
bool ArrowStopDelete(const long   chart_ID=0,       // identifiant du graphique
                     const string name="ArrowStop") // nom de l'étiquette
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- efface le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à supprimer le signe \"Stop\" ! Code d'erreur = ",GetLa
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées du point d'ancrage du signe
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points en dessinant le signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crée un signe Stop sur le graphique
   if(!ArrowStopCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplacez le point d'ancrage et changez sa position par rapport au si
//--- compteur de boucle
   int h_steps=bars*2/5;
//--- déplace le point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- déplace le point
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,025 secondes
      Sleep(25);
     }
//--- modifie l'emplacement du point d'ancrage par rapport au signal
   ArrowStopAnchorChange(0,InpName,ANCHOR_TOP);
//--- redessine le graphique
   ChartRedraw();
//--- compteur de boucle
   h_steps=bars*2/5;
//--- déplace le point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- déplace le point
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,025 secondes
      Sleep(25);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- efface le signe du graphique
   ArrowStopDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_CHECK

Signe coche.

Note

La position du point d'ancrage par rapport au signe peut être sélectionnée dans l'énumération ENUM_ARROW_ANCHOR.

Les gros signes (plus de 5) peuvent être créés simplement en définissant la valeur de la propriété OBJPROP_WIDTH correspondante lors de l'édition du code dans MetaEditor.

Exemple

Le script suivant crée et déplace le signe Coche sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine le signe \"Coche\"."
#property description "La coordonnée du point d'ancrage est définie en"
#property description "pourcentage de la taille de la fenêtre du graphique"
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="ArrowCheck"; // nom du signe
input int               InpDate=10;           // Date du point d'ancrage en %
input int               InpPrice=50;          // Prix du point d'ancrage en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // type d'ancrage
input color             InpColor=clrRed;      // Couleur du signe
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Style de la ligne de bordure
input int               InpWidth=5;           // Taille du signe
input bool              InpBack=false;        // Signe en arrière plan
input bool              InpSelection=false;   // Mise en surbrillance pour déplacer
input bool              InpHidden=true;       // Caché dans la liste des objets
input long              InpZOrder=0;          // Priorité du clic de souris
//+------------------------------------------------------------------------------
//| Crée un signal Coche
//+------------------------------------------------------------------------------
bool ArrowCheckCreate(const long               chart_ID=0,           // identifiant du
                      const string             name="ArrowCheck",    // nom du signe
                      const int                sub_window=0,         // indice de sous-
                      datetime                 time=0,               // heure du point
                      double                   price=0,              // prix du point c
                      const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // type d'ancrage
                      const color              clr=clrRed,           // Couleur du sigr
                      const ENUM_LINE_STYLE    style=STYLE_SOLID,    // Style de la lig
                      const int                width=3,              // Taille du signe
                      const bool               back=false,           // en arrière-plar
                      const bool               selection=true,       // mise en surbril
                      const bool               hidden=true,          // caché dans la l
                      const long               z_order=0)            // priorité du cli
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_CHECK,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer le signe \"Coche\"! Code d'erreur = ",GetLastEr
      return(false);
     }
//--- définit le type d'ancrage
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- définit la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (vrai) ou désactive (faux) le mode de déplacement du signal par la souris
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
bool ArrowCheckMove(const long   chart_ID=0,        // identifiant du graphique
                    const string name="ArrowCheck", // nom de l'objet
                    datetime     time=0,            // coordonnées du point d'ancrage
                    double       price=0)           // coordonnées du point d'ancrage
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie le type d'ancrage du signe Coche
//+------------------------------------------------------------------------------------
bool ArrowCheckAnchorChange(const long              chart_ID=0,        // identifiant
                            const string            name="ArrowCheck", // nom de l'obj
                            const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // type d'ancra
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le type d'ancrage
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le type d'ancrage ! Code d'erreur = ",GetLastEr
      return(false);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le signe Coche
//+------------------------------------------------------------------------------------
bool ArrowCheckDelete(const long   chart_ID=0,        // identifiant du graphique
                      const string name="ArrowCheck") // nom du signe
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- efface le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à supprimer le signe \"Check\"! Code d'erreur = ",GetLa
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées du point d'ancrage du signe
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points en dessinant le signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crée le signal Check sur le graphique
   if(!ArrowCheckCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplacez le point d'ancrage et changez sa position par rapport au si
//--- compteur de boucle
   int h_steps=bars*2/5;
//--- déplace le point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- déplace le point
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,025 secondes
      Sleep(25);
     }
//--- modifie l'emplacement du point d'ancrage par rapport au signal
   ArrowCheckAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redessine le graphique
   ChartRedraw();
//--- compteur de boucle
   h_steps=bars*2/5;
//--- déplace le point d'ancrage
   for(int i=0;i<h_steps;i++)
     {
      //--- utilise la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- déplace le point
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,025 secondes
      Sleep(25);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- efface le signe du graphique
   ArrowCheckDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_LEFT_PRICE

Etiquette de Prix à Gauche

Exemple

Le script suivant crée et déplace l'étiquette de prix à gauche sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée l'étiquette de prix à droite sur le graphique."
#property description "La coordonnée du point d'ancrage est définie en"
#property description "pourcentage de la taille de la fenêtre du graphique"
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="LeftPrice";  // Nom de l'étiquette de prix
input int               InpDate=100;          // Date du point d'ancrage en %
input int               InpPrice=10;          // Prix du point d'ancrage en %
input color             InpColor=clrRed;      // Couleur de l'étiquette de prix
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Style de la ligne de bordure
input int               InpWidth=2;           // Taille de l'étiquette de prix
input bool              InpBack=false;        // Etiquette en arrière-plan
input bool              InpSelection=true;    // Mise en surbrillance pour déplacer
input bool              InpHidden=true;       // Caché dans la liste des objets
input long              InpZOrder=0;          // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée l'étiquette de prix à gauche
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------------------------
bool ArrowLeftPriceCreate(const long            chart_ID=0,        // identifiant du g
                          const string          name="LeftPrice",  // nom de l'étiquet
                          const int             sub_window=0,      // indice de sous-f
                          datetime              time=0,            // heure du point d
                          double                price=0,           // prix du point d'
                          const color           clr=clrRed,        // couleur de l'éti
                          const ENUM_LINE_STYLE style=STYLE_SOLID, // Style de la ligr
                          const int             width=1,           // taille de l'étic
                          const bool            back=false,        // en arrière-plan
                          const bool            selection=true,    // mise en surbrill
                          const bool            hidden=true,       // caché dans la li
                          const long            z_order=0)         // priorité du clic
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée l'étiquette de prix
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_LEFT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer l'étiquette de prix à gauche ! Code d'erreur =
      return(false);
     }
//--- définit la couleur de l'étiquette
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille de l'étiquette
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de l'étiquette par la
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
bool ArrowLeftPriceMove(const long   chart_ID=0,       // identifiant du graphique
                        const string name="LeftPrice", // nom de l'étiquette
                        datetime     time=0,           // coordonnées du point d'ancra
                        double       price=0)          // coordonnées du point d'ancra
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime l'étiquette de prix à gauche du graphique
//+------------------------------------------------------------------------------------
bool ArrowLeftPriceDelete(const long   chart_ID=0,       // identifiant du graphique
                          const string name="LeftPrice") // nom de l'étiquette
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'étiquette
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à supprimer l'étiquette de prix à gauche ! Code d'erreu
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées du point d'ancrage de l'étiqu
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner l'étiquette
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crée l'étiquette de prix à gauche sur le graphique
   if(!ArrowLeftPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

```
      }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplace le point d'ancrage
//--- compteur de boucle
   int v_steps=accuracy*4/5;
//--- déplace le point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p<accuracy-1)
         p+=1;
      //--- déplace le point
      if(!ArrowLeftPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'étiquette depuis le graphique
   ArrowLeftPriceDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_RIGHT_PRICE

Etiquette de Prix à Droite

Exemple

Le script suivant crée et déplace la bonne étiquette de prix sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée une étiquette de prix à droite sur le graphique.
#property description "La coordonnée du point d'ancrage est définie en"
#property description "pourcentage de la taille de la fenêtre du graphique"
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="RightPrice"; // Nom de l'étiquette de prix
input int               InpDate=0;            // Date du point d'ancrage en %
input int               InpPrice=90;          // Prix du point d'ancrage en %
input color             InpColor=clrRed;      // Couleur de l'étiquette de prix
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Style de la ligne de bordure
input int               InpWidth=2;           // Taille de l'étiquette de prix
input bool              InpBack=false;        // Etiquette en arrière-plan
input bool              InpSelection=true;    // Mise en surbrillance pour déplacer
input bool              InpHidden=true;       // Caché dans la liste des objets
input long              InpZOrder=0;          // Priorité du clic de souris
//+------------------------------------------------------------------------------
//| Crée l'étiquette de prix à droite
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------------------------
bool ArrowRightPriceCreate(const long            chart_ID=0,        // identifiant du
                           const string          name="RightPrice", // nom de l'étique
                           const int             sub_window=0,      // indice de sous-
                           datetime              time=0,            // heure du point
                           double                price=0,           // prix du point c
                           const color           clr=clrRed,        // couleur de l'ét
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // Style de la lig
                           const int             width=1,           // taille de l'éti
                           const bool            back=false,        // en arrière-plan
                           const bool            selection=true,    // mise en surbril
                           const bool            hidden=true,       // caché dans la l
                           const long            z_order=0)         // priorité pour l
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée l'étiquette de prix
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_RIGHT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer l'étiquette de prix à droite ! Code d'erreur =
      return(false);
     }
//--- définit la couleur de l'étiquette
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille de l'étiquette
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de l'étiquette par la
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
bool ArrowRightPriceMove(const long   chart_ID=0,        // identifiant du graphique
                         const string name="RightPrice", // nom de l'étiquette
                         datetime     time=0,            // coordonnées du point d'anc
                         double       price=0)           // coordonnées du point d'anc
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le prix de l'étiquette à droite depuis le graphique
//+------------------------------------------------------------------------------------
bool ArrowRightPriceDelete(const long   chart_ID=0,        // identifiant du graphique
                           const string name="RightPrice") // nom de l'étiquette
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'étiquette
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": n'a pas réussi à créer l'étiquette de prix à droite ! Code d'erreur = '
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées du point d'ancrage de l'étiqu
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner l'étiquette
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crée le prix de l'étiquette à droite sur le graphique
   if(!ArrowRightPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant, déplace le point d'ancrage
//--- compteur de boucle
   int v_steps=accuracy*4/5;
//--- déplace le point d'ancrage
   for(int i=0;i<v_steps;i++)
     {
      //--- utilise la valeur suivante
      if(p>1)
         p-=1;
      //--- déplace le point
      if(!ArrowRightPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'étiquette depuis le graphique
   ArrowRightPriceDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_BUY

Signe d'achat.

Exemple

Le script suivant crée et déplace le signe d'Achat sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine des signes \"Achat\" sur le graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input color InpColor=C'3,95,172'; // Couleur des signaux
//+------------------------------------------------------------------------------------
//| Crée un signe d'Achat
//+------------------------------------------------------------------------------------
bool ArrowBuyCreate(const long            chart_ID=0,        // identifiant du graphic
                    const string          name="ArrowBuy",   // nom du signe
                    const int             sub_window=0,      // indice de sous-fenêtre
                    datetime              time=0,            // heure du point d'ancra
                    double                price=0,           // prix du point d'ancrag
                    const color           clr=C'3,95,172',   // Couleur du signe
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Style de trait (en su
                    const int             width=1,           // taille de la ligne (en
                    const bool            back=false,        // en arrière-plan
                    const bool            selection=false,   // mise en surbrillance p
```

© 2000-2024, MetaQuotes Ltd.

```
                    const bool            hidden=true,       // caché dans la liste de
                    const long            z_order=0)         // priorité du clic de so
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_BUY,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": n'a pas réussi à créer le signe d'\"Achat\" ! Code d'erreur = ",GetLast
      return(false);
     }
//--- définit la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit un style de ligne (lorsqu'elle est en surbrillance)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille de la ligne (lorsqu'elle est en surbrillance)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (vrai) ou désactive (faux) le mode de déplacement du signal par la souris
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
bool ArrowBuyMove(const long   chart_ID=0,       // identifiant du graphique
                  const string name="ArrowBuy",  // nom de l'objet
                  datetime     time=0,           // coordonnées du point d'ancrage temp
                  double       price=0)          // coordonnées du point d'ancrage prix
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
```

```
      {
       Print(__FUNCTION__,
             " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
       return(false);
      }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le signe d'Achat
//+------------------------------------------------------------------------------------
bool ArrowBuyDelete(const long   chart_ID=0,      // identifiant du graphique
                    const string name="ArrowBuy") // nom du signe
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- efface le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à supprimer le signe d'\"Achat\" sign! Code d'erreur =
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
   datetime date[]; // tableau pour stocker les dates des barres visibles
   double   low[];  // tableau pour stocker les prix Low des barres visibles
   double   high[]; // tableau pour stocker les prix High des barres visibles
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("N'a pas réussi à copier les valeurs des prix Low ! Code d'erreur = ",GetL
      return;
     }
//--- remplit le tableau des prix High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("N'a pas réussi à copier les valeurs des prix High ! Code d'erreur = ",Get
      return;
     }
//--- crée des signaux d'Achat en point Low pour chaque barre visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyCreate(0,"ArrowBuy_"+(string)i,0,date[i],low[i],InpColor))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- modifie des signaux d'Achat en point High pour chaque barre visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyMove(0,"ArrowBuy_"+(string)i,date[i],high[i]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- efface le signe Achat
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyDelete(0,"ArrowBuy_"+(string)i))
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_SELL

Signe de Vente.

Exemple

Le script suivant crée et déplace le signal Sell sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine des signes \"Vente\" sur le graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input color InpColor=C'225,68,29'; // Couleur des signaux
//+------------------------------------------------------------------------------------
//| Crée un signe de Vente
//+------------------------------------------------------------------------------------
bool ArrowSellCreate(const long            chart_ID=0,        // identifiant du graphi
                     const string          name="ArrowSell",  // nom du signe
                     const int             sub_window=0,      // indice de sous-fenêt
                     datetime              time=0,            // heure du point d'anc
                     double                price=0,           // prix du point d'ancra
                     const color           clr=C'225,68,29',  // Couleur du signe
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // Style de trait (en su
                     const int             width=1,           // taille de la ligne (e
                     const bool            back=false,        // en arrière-plan
                     const bool            selection=false,   // mise en surbrillance
```

© 2000-2024, MetaQuotes Ltd.

```
                      const bool            hidden=true,       // caché dans la liste d
                      const long            z_order=0)         // priorité du clic de s
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_SELL,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer le signe de \"Vente\" ! Code d'erreur = ",GetLa
      return(false);
     }
//--- définit la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit un style de ligne (lorsqu'elle est en surbrillance)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille de la ligne (lorsqu'elle est en surbrillance)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (vrai) ou désactive (faux) le mode de déplacement du signal par la souris
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des d
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
bool ArrowSellMove(const long   chart_ID=0,       // identifiant du graphique
                   const string name="ArrowSell", // nom de l'objet
                   datetime     time=0,           // coordonnées du point d'ancrage te
                   double       price=0)          // coordonnées du point d'ancrage pr
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
```

```
      {
       Print(__FUNCTION__,
             " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
       return(false);
      }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le signal de Vente
//+------------------------------------------------------------------------------------
bool ArrowSellDelete(const long   chart_ID=0,       // identifiant du graphique
                     const string name="ArrowSell") // nom du signe
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- efface le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": n'a pas réussi à supprimer le signe de \"Vente\" ! Code d'erreur = ",Ge
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
   datetime date[]; // tableau pour stocker les dates des barres visibles
   double   low[];  // tableau pour stocker les prix Low des barres visibles
   double   high[]; // tableau pour stocker les prix High des barres visibles
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("N'a pas réussi à copier les valeurs des prix Low ! Code d'erreur = ",GetI
      return;
     }
//--- remplit le tableau des prix High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("N'a pas réussi à copier les valeurs des prix High ! Code d'erreur = ",Get
      return;
     }
//--- crée des signaux de Vente en point High pour chaque barre visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellCreate(0,"ArrowSell_"+(string)i,0,date[i],high[i],InpColor))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- modifie des signaux de Vente en point Low pour chaque barre visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellMove(0,"ArrowSell_"+(string)i,date[i],low[i]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- supprime les signaux de Vente
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellDelete(0,"ArrowSell_"+(string)i))
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW

Objet flèche.

Note

La position du point d'ancrage par rapport à l'objet peut être sélectionnée dans ENUM_ARROW_ANCHOR.

Les grandes flèches (plus de 5) ne peuvent être créées qu'en définissant la valeur appropriée de la propriété OBJPROP_WIDTH lors de l'écriture du code dans MetaEditor.

Le type de flèche désiré peut être sélectionné en utilisant l'un des codes de symbole de la police de caractères Wingdings.

Exemple

Le script suivant crée l'objet Flèche sur le graphique et modifie son type. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée une flèche aléatoire dans la fenêtre du graphiqu
#property description "La coordonnée du point d'ancrage est définie en"
#property description "pourcentage de la taille de la fenêtre du graphique"
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="Arrow";        // Nom de la flêche
input int               InpDate=50;             // Date du point d'ancrage en %
```

© 2000-2024, MetaQuotes Ltd.

```
input int                InpPrice=50;            // Prix du point d'ancrage en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP;    // Type d'ancrage
input color              InpColor=clrDodgerBlue; // Couleur de la flêche
input ENUM_LINE_STYLE    InpStyle=STYLE_SOLID;   // Style de la ligne de bordure
input int                InpWidth=10;            // Taille de la flêche
input bool               InpBack=false;          // Flêche en arrière-plan
input bool               InpSelection=false;     // Mise en surbrillance pour déplacer
input bool               InpHidden=true;         // Caché dans la liste des objets
input long               InpZOrder=0;            // Priorité pour les clics de souris
//+------------------------------------------------------------------------------------
//| Crée la flèche
//+------------------------------------------------------------------------------------
bool ArrowCreate(const long              chart_ID=0,           // identifiant du graph
                 const string            name="Arrow",         // nom de la flêche
                 const int               sub_window=0,         // indice de sous-fenêt
                 datetime                time=0,               // Heure du point d'anc
                 double                  price=0,              // Prix du point d'ancr
                 const uchar             arrow_code=252,       // code de la flêche
                 const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // position du point d'
                 const color             clr=clrRed,           // couleur de la flêche
                 const ENUM_LINE_STYLE   style=STYLE_SOLID,    // Style de la ligne de
                 const int               width=3,              // taille de la flêche
                 const bool              back=false,           // en arrière-plan
                 const bool              selection=true,       // mise en surbrillance
                 const bool              hidden=true,          // Caché dans la liste
                 const long              z_order=0)            // priorité des clics c
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeArrowEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée une flèche
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": n'a pas réussi à créer une flèche ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- Définit le code de la flèche
   ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,arrow_code);
//--- définit le type d'ancrage
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- Définit la couleur de la flèche
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Définit la taille de la flèche
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (vrai) ou désactive (faux) le mode de déplacement de la flèche par la sou
//--- lors de la création d'un objet graphique utilisant la fonction ObjectCreate, l'o
//--- mis en surbrillance et déplacé par défaut. Dans cette méthode, le paramètre de s
//--- est vrai par défaut, permettant de mettre en surbrillance et de déplacer l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
bool ArrowMove(const long   chart_ID=0,   // identifiant du graphique
               const string name="Arrow", // nom de l'objet
               datetime     time=0,       // coordonnées du point d'ancrage temporel
               double       price=0)      // coordonnées du point d'ancrage prix
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie le code de la flèche
//+------------------------------------------------------------------------------------
bool ArrowCodeChange(const long   chart_ID=0,   // identifiant du graphique
                     const string name="Arrow", // nom de l'objet
                     const uchar  code=252)     // code de la flèche
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le code de la flèche
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,code))
     {
      Print(__FUNCTION__,
            ": impossible de modifier le code de la flêche ! Code d'erreur = ",GetLast
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie le type d'ancrage
//+------------------------------------------------------------------------------------
bool ArrowAnchorChange(const long              chart_ID=0,        // identifiant du gr
                       const string            name="Arrow",      // nom de l'objet
                       const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // type d'ancrage
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le type d'ancrage
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le type d'ancrage ! Code d'erreur = ",GetLastEr
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime la flêche
//+------------------------------------------------------------------------------------
bool ArrowDelete(const long   chart_ID=0,   // identifiant du graphique
                 const string name="Arrow") // nom de la flêche
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime la flêche
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": n'a pas réussi à supprimer la flêche ! Code d'erreur = ",GetLastError()
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------------------------
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- taille du tableau des prix
   int accuracy=1000;
//--- tableaux pour stocker les valeurs de date et de prix à utiliser
//--- pour le réglage et la modification des coordonnées du point d'ancrage du signe
   datetime date[];
   double   price[];
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix
//--- trouve les valeurs les plus élevées et les plus faibles du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définit un changement de prix et remplit le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définit les points pour dessiner la flèche
   int d=InpDate*(bars-1)/100;
```

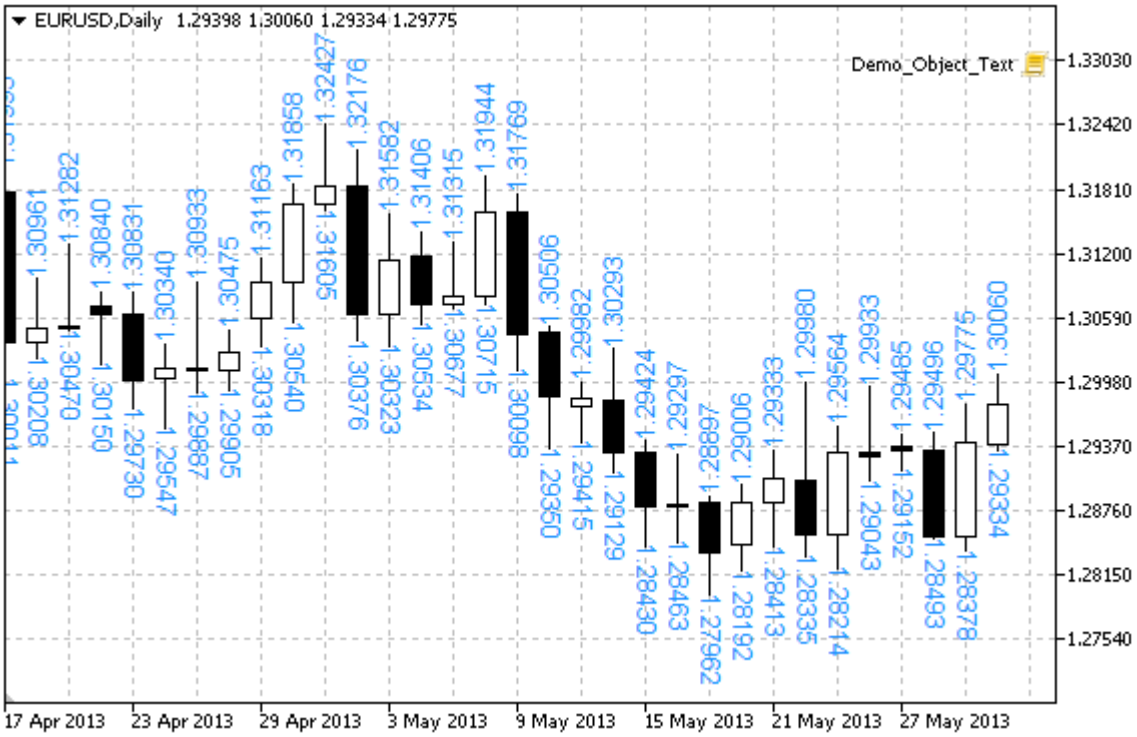
© 2000-2024, MetaQuotes Ltd.

```
   int p=InpPrice*(accuracy-1)/100;
//--- crée une flèche sur le graphique
   if(!ArrowCreate(0,InpName,0,date[d],price[p],32,InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessine le graphique
   ChartRedraw();
//--- considére tous les cas de création de flèches dans la boucle
   for(int i=33;i<256;i++)
     {
      if(!ArrowCodeChange(0,InpName,(uchar)i))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai d'une demie seconde
      Sleep(500);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- efface la flèche du graphique
   ArrowDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_TEXT

Objet Texte.



Note

La position du point d'ancrage relative au texte peut être sélectionné dans l'énumération ENUM_ANCHOR_POINT. Vous pouvez également changer l'angle de pente du texte en utilisant la propriété OBJPROP_ANGLE.

Exemple

Le script suivant crée plusieurs objets Texte sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un objet graphique \"Texte\"."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpFont="Arial";          // Police de caractères
input int               InpFontSize=10;           // Taille de la police de caractères
input color             InpColor=clrRed;          // Couleur
input double            InpAngle=90.0;            // Angle de pente en degrés
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_LEFT;    // type d'ancrage
input bool              InpBack=false;            // Objet en arrière-plan
input bool              InpSelection=false;       // Mise en surbrillance pour déplace
input bool              InpHidden=true;           // Caché dans la liste des objets
```

© 2000-2024, MetaQuotes Ltd.

```
input long              InpZOrder=0;             // Priorité du clic de souris
//+------------------------------------------------------------------------------
//| Création de l'objet Texte
//+------------------------------------------------------------------------------
bool TextCreate(const long              chart_ID=0,               // identifiant du gr
                const string            name="Text",              // nom de l'objet
                const int               sub_window=0,             // indice de sous-fe
                datetime                time=0,                   // heure du point d'
                double                  price=0,                  // prix du point d'a
                const string            text="Text",              // le texte lui-même
                const string            font="Arial",             // police de caracté
                const int               font_size=10,             // taille de la poli
                const color             clr=clrRed,               // couleur
                const double            angle=0.0,                // pente du texte
                const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // type d'ancrage
                const bool              back=false,               // en arrière plan
                const bool              selection=false,          // mise en surbrilla
                const bool              hidden=true,              // caché dans la lis
                const long              z_order=0)                // priorité pour le
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeTextEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée l'objet Texte
   if(!ObjectCreate(chart_ID,name,OBJ_TEXT,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de créer l'objet \"Texte\" ! Code d'erreur = ",GetLastErro
      return(false);
     }
//--- définit le texte
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- définit la police de caractères du texte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- définit la taille de la police de caractères
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- définit l'angle de pente du texte
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- définit le type d'ancrage
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- définit la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de l'objet avec la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le point d'ancrage
//+------------------------------------------------------------------------------------
bool TextMove(const long   chart_ID=0,  // identifiant du graphique
              const string name="Text", // nom de l'objet
              datetime     time=0,      // coordonnées du point d'ancrage temporel
              double       price=0)     // coordonnées du point d'ancrage prix
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie le texte de l'objet
//+------------------------------------------------------------------------------------
bool TextChange(const long   chart_ID=0,  // identifiant du graphique
                const string name="Text", // nom de l'objet
                const string text="Text") // texte
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le texte de l'objet
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le texte ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------------------------
//| Supprime l'objet Texte
//+------------------------------------------------------------------------------------
bool TextDelete(const long   chart_ID=0,  // identifiant du graphique
                const string name="Text") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : impossible de créer l'objet \"Texte\" ! Code d'erreur = ",GetLastErro
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeTextEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
   datetime date[]; // tableau pour stocker les dates des barres visibles
   double   low[];  // tableau pour stocker les prix Low des barres visibles
   double   high[]; // tableau pour stocker les prix High des barres visibles
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
       return;
      }
//--- remplit le tableau des prix Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
      {
       Print("N'a pas réussi à copier les valeurs des prix Low ! Code d'erreur = ",GetI
       return;
      }
//--- remplit le tableau des prix High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
      {
       Print("N'a pas réussi à copier les valeurs des prix High ! Code d'erreur = ",Get
       return;
      }
//--- définit à quelle fréquence les textes doivent être affichés
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- définit le pas
   int step=1;
   switch(scale)
      {
       case 0:
          step=12;
          break;
       case 1:
          step=6;
          break;
       case 2:
          step=4;
          break;
       case 3:
          step=2;
          break;
      }
//--- crée les textes pour les valeurs High et Low des barres (avec les gaps)
   for(int i=0;i<bars;i+=step)
      {
       //--- crée les textes
       if(!TextCreate(0,"TextHigh_"+(string)i,0,date[i],high[i],DoubleToString(high[i],
          InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
       if(!TextCreate(0,"TextLow_"+(string)i,0,date[i],low[i],DoubleToString(low[i],5),
          InpColor,-InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- supprime les textes
   for(int i=0;i<bars;i+=step)
     {
      if(!TextDelete(0,"TextHigh_"+(string)i))
         return;
      if(!TextDelete(0,"TextLow_"+(string)i))
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_LABEL

Objet Etiquette.

Note

La position du point d'ancrage par rapport à l'étiquette peut être sélectionnée dans l'énumération ENUM_ANCHOR_POINT. Les coordonnées du point d'ancrage sont définies en pixels.

Vous pouvez également sélectionner le coin d'ancrage de l'étiquette parmi l'ENUM_BASE_CORNER énumération.

Exemple

Le script suivant crée et déplace l'objet Zone d'Edition sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un objet graphique \"Etiquette\"."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="Label";          // Nom de l'étiquette
input int               InpX=150;                 // distance de l'axe X
input int               InpY=150;                 // distance de l'axe Y
input string            InpFont="Arial";          // Police de caractères
input int               InpFontSize=14;           // Taille de la police de caractères
input color             InpColor=clrRed;          // Couleur
input double            InpAngle=0.0;             // Angle de la pente en degrés
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // type d'ancrage
input bool              InpBack=false;           // Objet en arrière-plan
input bool              InpSelection=true;       // Mise en surbrillance pour déplacer
input bool              InpHidden=true;          // Caché dans la liste des objets
input long              InpZOrder=0;             // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée une étiquette
//+------------------------------------------------------------------------------------
bool LabelCreate(const long              chart_ID=0,               // identifiant du g
                 const string            name="Label",             // nom de l'étiquet
                 const int               sub_window=0,             // indice de sous-f
                 const int               x=0,                      // coordonnée X
                 const int               y=0,                      // coordonnée Y
                 const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // coin d'ancrage d
                 const string            text="Label",             // texte
                 const string            font="Arial",             // police de caract
                 const int               font_size=10,             // taille de la pol
                 const color             clr=clrRed,               // couleur
                 const double            angle=0.0,                // pente du texte
                 const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // type d'ancrage
                 const bool              back=false,               // en arrière-plan
                 const bool              selection=false,          // mise en surbrill
                 const bool              hidden=true,              // caché dans la li
                 const long              z_order=0)                // priorité du clic
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée une étiquette
   if(!ObjectCreate(chart_ID,name,OBJ_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": impossible de créer l'étiquette ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- définit les coordonnées de l'étiquette
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- définit le coin du graphique, relativement auquel les coordonnées du point sont
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- définit le texte
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- définit la police de caractères du texte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- définit la taille de la police de caractères
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- définit l'angle de pente du texte
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- définit le type d'ancrage
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de l'étiquette par la
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des c
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace l'étiquette
//+------------------------------------------------------------------------------------
bool LabelMove(const long   chart_ID=0,   // identifiant du graphique
               const string name="Label", // nom de l'étiquette
               const int    x=0,          // coordonnée X
               const int    y=0)          // Coordonnée Y
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace l'étiquette
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la coordonnée X de l'étiquette ! Code d'erreur
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la coordonnée Y de l'étiquette ! Code d'erreur
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change le coin du graphique pour lier l'étiquette
//+------------------------------------------------------------------------------------
bool LabelChangeCorner(const long             chart_ID=0,               // identifiant
                       const string           name="Label",             // nom de l'ét
                       const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // coin d'anc
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
```

```
//--- modifie le coin d'ancrage
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": impossible de modifier le coin d'ancrage ! Code d'erreur = ",GetLastEr
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie le texte de l'étiquette texte
//+------------------------------------------------------------------------------------
bool LabelTextChange(const long   chart_ID=0,   // identifiant du graphique
                     const string name="Label", // nom de l'objet
                     const string text="Text")  // texte
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le texte de l'objet
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le texte ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime l'étiquette
//+------------------------------------------------------------------------------------
bool LabelDelete(const long   chart_ID=0,   // identifiant du graphique
                 const string name="Label") // nom de l'étiquette
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'étiquette
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": impossible de supprimer l'étiquette ! Code d'erreur = ",GetLastError())
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- stocke les coordonnées de l'étiquette dans les variables locales
   int x=InpX;
   int y=InpY;
//--- taille de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définit la taille de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Impossible d'obtenir la largeur du graphique ! Code d'erreur = ",GetLastE
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Impossible d'obtenir la hauteur du graphique ! Code d'erreur = ",GetLastE
      return;
     }
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpX<0 || InpX>x_distance-1 || InpY<0 || InpY>y_distance-1)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- prépare le texte initial de l'étiquette
   string text;
   StringConcatenate(text,"Coin supérieur gauche : ",x,",",y);
//--- crée l'étiquette texte sur le graphique
   if(!LabelCreate(0,InpName,0,InpX,InpY,CORNER_LEFT_UPPER,text,InpFont,InpFontSize,
      InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessine le graphique et attend 1 demie seconde
   ChartRedraw();
   Sleep(500);
//--- déplace l'étiquette et change son texte simultanément
//--- nombre d'itérations par axes
   int h_steps=(int)(x_distance/2-InpX);
   int v_steps=(int)(y_distance/2-InpY);
//--- déplace l'étiquette vers le bas
   for(int i=0;i<v_steps;i++)
     {
      //--- modifie la coordonnée
      y+=2;
      //--- déplace l'étiquette et modifie son texte
      MoveAndTextChange(x,y,"Coin supérieur gauche : ");
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- délai d'une demie seconde
   Sleep(500);
//--- déplace l'étiquette à droite
   for(int i=0;i<h_steps;i++)
     {
      //--- modifie la coordonnée
      x+=2;
      //--- déplace l'étiquette et modifie son texte
      MoveAndTextChange(x,y,"Coin supérieur gauche : ");
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- déplace l'étiquette vers le haut
   for(int i=0;i<v_steps;i++)
     {
      //--- modifie la coordonnée
      y-=2;
      //--- déplace l'étiquette et modifie son texte
      MoveAndTextChange(x,y,"Coin supérieur gauche : ");
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- déplace l'étiquette vers la gauche
   for(int i=0;i<h_steps;i++)
     {
      //--- modifie la coordonnée
      x-=2;
      //--- déplace l'étiquette et modifie son texte
      MoveAndTextChange(x,y,"Coin supérieur gauche : ");
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- déplace maintenant le point en changeant le coin d'ancrage
//--- le déplace vers le coin inférieur gauche
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_LOWER))
      return;
//--- modifie l'étiquette texte
   StringConcatenate(text,"Coin inférieur gauche : ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redessine le graphique et attend 2 secondes
   ChartRedraw();
   Sleep(2000);
//--- déplace vers le coin inférieur gauche
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_LOWER))
      return;
//--- modifie l'étiquette texte
   StringConcatenate(text,"Coin inférieur gauche : ",x,",",y);
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redessine le graphique et attend 2 secondes
   ChartRedraw();
   Sleep(2000);
//--- déplace vers le coin supérieur droit
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_UPPER))
      return;
//--- modifie l'étiquette texte
   StringConcatenate(text,"Coin supérieur droit : ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redessine le graphique et attend 2 secondes
   ChartRedraw();
   Sleep(2000);
//--- déplace vers le coin supérieur gauche
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_UPPER))
      return;
//--- modifie l'étiquette texte
   StringConcatenate(text,"Coin supérieur gauche : ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redessine le graphique et attend 2 secondes
   ChartRedraw();
   Sleep(2000);
//--- supprime l'étiquette
   LabelDelete(0,InpName);
//--- redessine le graphique et attend 1 demie seconde
   ChartRedraw();
   Sleep(500);
//---
  }
//+------------------------------------------------------------------------------------+
//| La fonction déplace l'objet et modifie son texte                                   |
//+------------------------------------------------------------------------------------+
bool MoveAndTextChange(const int x,const int y,string text)
  {
//--- déplace l'étiquette
   if(!LabelMove(0,InpName,x,y))
      return(false);
//--- modifie l'étiquette texte
   StringConcatenate(text,text,x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return(false);
//--- vérifie si l'exécution du script a été stoppée
   if(IsStopped())
      return(false);
//--- redessine le graphique
   ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
// délai de 0.01 seconde
   Sleep(10);
//--- sort de la fonction
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_BUTTON

Objet bouton.

Note

Les coordonnées du point d'ancrage sont définies en pixels. Vous pouvez sélectionner le coin d'ancrage du bouton dans ENUM_BASE_CORNER.

Exemple

Le script suivant crée et déplace l'objet Bouton sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée un bouton sur le graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string           InpName="Button";            // Nom du bouton
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Coin du graphique pour l'ancrag
input string           InpFont="Arial";             // Police de caractères
input int              InpFontSize=14;              // Taille de la police de caractèr
input color            InpColor=clrBlack;           // Couleur du texte
input color            InpBackColor=C'236,233,216'; // Couleur de l'arrière-plan
input color            InpBorderColor=clrNONE;      // couleur de la bordure
input bool             InpState=false;              // Appuyé/Relâché
input bool             InpBack=false;               // Objet en arrière-plan
input bool             InpSelection=false;          // Mise en surbrillance pour dépla
```

© 2000-2024, MetaQuotes Ltd.

```
input bool              InpHidden=true;              // Caché dans la liste des objets
input long              InpZOrder=0;                 // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée le bouton
//+------------------------------------------------------------------------------------
bool ButtonCreate(const long              chart_ID=0,                // identifiant du
                  const string            name="Button",             // nom du bouton
                  const int               sub_window=0,              // indice de sous-
                  const int               x=0,                       // coordonnée X
                  const int               y=0,                       // coordonnée Y
                  const int               width=50,                  // largeur du bout
                  const int               height=18,                 // hauteur du bout
                  const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER,  // coin du graphic
                  const string            text="Button",             // texte
                  const string            font="Arial",              // police de carac
                  const int               font_size=10,              // taille de la po
                  const color             clr=clrBlack,              // couleur du text
                  const color             back_clr=C'236,233,216',   // couleur en arri
                  const color             border_clr=clrNONE,        // couleur de la b
                  const bool              state=false,               // Appuyé/Relâché
                  const bool              back=false,                // en arrière-plan
mise en surbrillance pour le déplacement                  const bool              sele
                  const bool              hidden=true,               // caché dans la l
                  const long              z_order=0)                 // priorité du cli
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée le bouton
   if(!ObjectCreate(chart_ID,name,OBJ_BUTTON,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": imposible de créer le bouton ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- définit les coordonnées du bouton
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- définit la taille du bouton
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- définit le coin du graphique, relativement auquel les coordonnées du point sont
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- définit le texte
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- définit la police de caractères du texte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- définit la taille de la police de caractères
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- définit la couleur du texte
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit la couleur d'arrière-plan
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- définit la couleur de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- définit l'état du bouton
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- active (true) ou désactive (false) le mode de déplacement du bouton à la souris
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace le bouton
//+------------------------------------------------------------------------------------
bool ButtonMove(const long   chart_ID=0,    // identifiant du graphique
                const string name="Button", // nom du bouton
                const int    x=0,           // coordonnée X
                const int    y=0)           // coordonnée Y
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le bouton
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la coordonnée X du bouton ! Code d'erreur = ",G
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la coordonnées Y du bouton ! Code d'erreur = ",
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie la taille du bouton
//+------------------------------------------------------------------------------------
bool ButtonChangeSize(const long   chart_ID=0,    // identifiant du graphique
```

© 2000-2024, MetaQuotes Ltd.

```
                        const string name="Button", // nom du bouton
                        const int    width=50,      // largeur du bouton
                        const int    height=18)     // hauteur du bouton
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie la taille du bouton
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            " : impossible de modifier la largeur du bouton ! Code d'erreur = ",GetLas
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            " : impossible de modifier la hauteur du bouton ! Code d'erreur = ",GetLas
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change le coin du graphique pour lier le bouton
//+------------------------------------------------------------------------------------
bool ButtonChangeCorner(const long             chart_ID=0,               // identifiar
                        const string           name="Button",            // nom du bou
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // coin du gr
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le coin d'ancrage
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": impossible de modifier le coin d'ancrage ! Code d'erreur = ",GetLastErr
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change le texte du bouton
//+------------------------------------------------------------------------------------
bool ButtonTextChange(const long   chart_ID=0,    // identifiant du graphique
                      const string name="Button", // nom du bouton
                      const string text="Text")   // texte
  {
//--- réinitialise la valeur d'erreur
```

© 2000-2024, MetaQuotes Ltd.

```
   ResetLastError();
//--- modifie le texte de l'objet
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le texte ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime le bouton
//+------------------------------------------------------------------------------------
bool ButtonDelete(const long   chart_ID=0,    // identifiant du graphique
                  const string name="Button") // nom du bouton
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime le bouton
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : impossible de supprimer le bouton ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- taille de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définit la taille de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Impossible d'obtenir la largeur du graphique ! Code d'erreur = ",GetLastE
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Impossible d'obtenir la hauteur du graphique ! Code d'erreur = ",GetLastE
      return;
     }
//--- définit le pas de changement de la taille du bouton
```

© 2000-2024, MetaQuotes Ltd.

```
   int x_step=(int)x_distance/32;
   int y_step=(int)y_distance/32;
//--- définit les coordonnées du bouton et sa taille
   int x=(int)x_distance/32;
   int y=(int)y_distance/32;
   int x_size=(int)x_distance*15/16;
   int y_size=(int)y_distance*15/16;
//--- crée le bouton
   if(!ButtonCreate(0,InpName,0,x,y,x_size,y_size,InpCorner,"Press",InpFont,InpFontSiz
      InpColor,InpBackColor,InpBorderColor,InpState,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
//--- redessine le graphique
   ChartRedraw();
//--- réduit le bouton dans la boucle
   int i=0;
   while(i<13)
     {
      //--- délai d'une demie seconde
      Sleep(500);
      //--- bascule le bouton à l'état appuyé
      ObjectSetInteger(0,InpName,OBJPROP_STATE,true);
      //--- redessine le graphique et attends 0,2 seconde
      ChartRedraw();
      Sleep(200);
      //--- redéfinit les coordonnées et la taille du bouton
      x+=x_step;
      y+=y_step;
      x_size-=x_step*2;
      y_size-=y_step*2;
      //--- réduit le bouton
      ButtonMove(0,InpName,x,y);
      ButtonChangeSize(0,InpName,x_size,y_size);
      //--- remet le bouton à l'état relâché
      ObjectSetInteger(0,InpName,OBJPROP_STATE,false);
      //--- redessine le graphique
      ChartRedraw();
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- incrémente le compteur de boucle
      i++;
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- supprime le bouton
   ButtonDelete(0,InpName);
   ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- attente d'1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_CHART

Objet Graphique.

Note

L'objet de type "OBJ_CHART" n'est pas supporté (non affiché) lors d'un test visuel.

Les coordonnées du point d'ancrage sont définies en pixels. Vous pouvez sélectionner le coin d'ancrage avec l'énumération ENUM_BASE_CORNER.

Le symbole, la période et l'échelle peuvent être sélectionnés pour l'objet Graphique. Les modes d'affichage des échelles des prix et des dates peuvent également être activés/désactivés.

Exemple

Le script suivant crée et déplace l'objet Graphique sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée un objet \"Graphique\"."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string           InpName="Chart";             // nom de l'objet
input string           InpSymbol="EURUSD";          // Symbole
input ENUM_TIMEFRAMES  InpPeriod=PERIOD_H1;         // Période
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Coin d'ancrage
input int              InpScale=2;                  //Echelle
```

© 2000-2024, MetaQuotes Ltd.

```
input bool              InpDateScale=true;           // Affichage de l'échelle des heu
input bool              InpPriceScale=true;          // Affichage de l'échelle des pri
input color             InpColor=clrRed;             // Couleur de la bordure au moment
input ENUM_LINE_STYLE   InpStyle=STYLE_DASHDOTDOT;   // Style de ligne lors qu'il est r
input int               InpPointWidth=1;             // Taille du point pour le déplace
input bool              InpBack=false;               // Objet en arrière-plan
input bool              InpSelection=true;           // Mise en surbrillance pour dépla
input bool              InpHidden=true;              // Caché dans la liste des objets
input long              InpZOrder=0;                 // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Création de l'objet Graphique
//+------------------------------------------------------------------------------------
bool ObjectChartCreate(const long              chart_ID=0,                // identifiar
                       const string            name="Chart",              // nom de l'o
                       const int               sub_window=0,              // indice de
                       const string            symbol="EURUSD",           // symbole
                       const ENUM_TIMEFRAMES   period=PERIOD_H1,          // période
                       const int               x=0,                       // coordonnée
                       const int               y=0,                       // coordonnée
                       const int               width=300,                 // largeur
                       const int               height=200,                // hauteur
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER,  // coin d'anc
                       const int               scale=2,                   // échelle
                       const bool              date_scale=true,           // Affichage
                       const bool              price_scale=true,          // Affichage
                       const color             clr=clrRed,                // couleur de
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,         // Style de t
                       const int               point_width=1,             // déplace la
                       const bool              back=false,                // en arrière
                       const bool              selection=false,           // mise en su
                       const bool              hidden=true,               // caché dans
                       const long              z_order=0)                 // priorité d
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée l'objet Graphique
   if(!ObjectCreate(chart_ID,name,OBJ_CHART,sub_window,0,0))
     {
      Print(__FUNCTION__,
            " : échec de création del'objet \"Graphique\" ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- définit les coordonnées de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- définit la taille de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- définit le coin du graphique, relativement auquel les coordonnées du point sont
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- définit le symbole
   ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol);
//--- définit la période
   ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period);
//--- définit l'échelle
   ObjectSetInteger(chart_ID,name,OBJPROP_CHART_SCALE,scale);
//--- affiche (true) ou cache (false) l'échelle des heures
   ObjectSetInteger(chart_ID,name,OBJPROP_DATE_SCALE,date_scale);
//--- affiche (true) ou cache (false) l'échelle des prix
   ObjectSetInteger(chart_ID,name,OBJPROP_PRICE_SCALE,price_scale);
//--- définit la couleur de la bordure lorsque le mode de mise en surbrillance de l'ob
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure lorsque le mode de mise en surbrillance
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille du point d'ancrage pour déplacer un objet
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de l'étiquette par la
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Définit le symbole et la période de l'objet Graphique
//+------------------------------------------------------------------------------------
bool ObjectChartSetSymbolAndPeriod(const long            chart_ID=0,       // identifi
                                   const string          name="Chart",     // nom de l
                                   const string          symbol="EURUSD",  // symbole
                                   const ENUM_TIMEFRAMES period=PERIOD_H1) // période
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit le symbole et la période de l'objet Graphique
   if(!ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol))
     {
      Print(__FUNCTION__,
            " : échec de sélection du symbole pour l'objet \"Graphique\" ! Code d'erre
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,période))
     {
      Print(__FUNCTION__,
```

```
            ": échec de sélection de la période pour l'objet \"Graphique\" ! Code d'er
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace l'objet Graphique
//+------------------------------------------------------------------------------------
bool ObjectChartMove(const long   chart_ID=0,   // identifiant du graphique (not Chart
                     const string name="Chart", // nom de l'objet
                     const int    x=0,          // Coordonnée X
                     const int    y=0)          // Coordonnée Y
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            " : échec du déplacement de la coordonnée X de l'objet \"Graphique\" ! Coo
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            " : échec du déplacement de la coordonnée Y de l'objet \"Graphique\" ! Coo
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change la taille de l'objet Graphique
//+------------------------------------------------------------------------------------
bool ObjectChartChangeSize(const long   chart_ID=0,   // identifiant du graphique (not
                           const string name="Chart", // nom de l'objet
                           const int    width=300,    // largeur
                           const int    height=200)   // hauteur
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie la taille de l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            " : échec du changement de largeur de l'objet \"Graphique\" ! Code d'erreu
      return(false);
     }
```

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            " : échec du changement de hauteur de l'objet \"Graphique\" ! Code d'erreu
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Retourne l'identifiant de l'objet Graphique
//+------------------------------------------------------------------------------------
long ObjectChartGetID(const long   chart_ID=0,   // identifiant du graphique (not Char
                      const string name="Chart") // nom de l'objet
  {
//--- prépare la variable pour récupérer l'identifiant de l'objet Graphique
   long id=-1;
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère l'identifiant
   if(!ObjectGetInteger(chart_ID,name,OBJPROP_CHART_ID,0,id))
     {
      Print(__FUNCTION__,
            " : échec de récupération de l'identifiant de l'objet \"Graphique\" ! Code
     }
//--- retourne le résultat
   return(id);
  }
//+------------------------------------------------------------------------------------
//| Supprime l'objet Graphique
//+------------------------------------------------------------------------------------
bool ObjectChartDelete(const long   chart_ID=0,   // identifiant du graphique (not Cha
                       const string name="Chart") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime le bouton
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de la suppression de l'objet \"Graphique\" ! Code d'erreur = ",G
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- récupère le nombre de symboles dans le Market Watch
   int  symbols=SymbolsTotal(true);
//--- vérife si le symbole ayant le nom spécifié est présent dans la liste des symbole
   bool exist=false;
   for(int i=0;i<symbols;i++)
      if(InpSymbol==SymbolName(i,true))
        {
         exist=true;
         break;
        }
   if(!exist)
     {
      Print("Erreur ! Le symbole ",InpSymbol," n'existe pas dans le \"Market Watch\" 
      return;
     }
//--- vérifie la validité des paramètres d'entrée
   if(InpScale<0 || InpScale>5)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }

//--- taille de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définit la taille de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Impossible d'obtenir la largeur du graphique ! Code d'erreur = ",GetLastE
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Impossible d'obtenir la hauteur du graphique ! Code d'erreur = ",GetLastE
      return;
     }
//--- définit les coordonnées et la taille de l'objet Graphique
   int x=(int)x_distance/16;
   int y=(int)y_distance/16;
   int x_size=(int)x_distance*7/16;
   int y_size=(int)y_distance*7/16;
//--- crée l'objet Graphique
   if(!ObjectChartCreate(0,InpName,0,InpSymbol,InpPeriod,x,y,x_size,y_size,InpCorner,I
      InpPriceScale,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
```

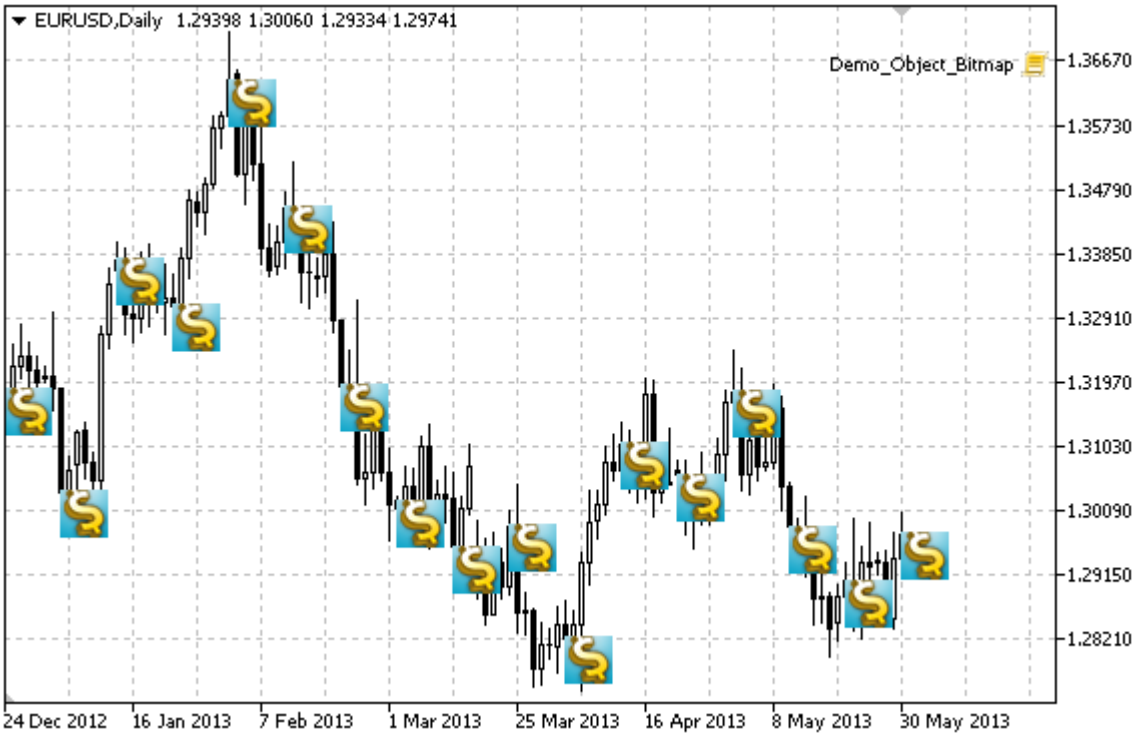
© 2000-2024, MetaQuotes Ltd.

```
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- élargit l'objet Graphique
   int steps=(int)MathMin(x_distance*7/16,y_distance*7/16);
   for(int i=0;i<steps;i++)
     {
      //--- redimmensionnement
      x_size+=1;
      y_size+=1;
      if(!ObjectChartChangeSize(0,InpName,x_size,y_size))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique et attend 0.01 seconde
      ChartRedraw();
      Sleep(10);
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- change la période du graphique
   if(!ObjectChartSetSymbolAndPeriod(0,InpName,InpSymbol,PERIOD_M1))
      return;
   ChartRedraw();
//--- trois secondes de délai
   Sleep(3000);
//--- supprime l'objet
   ObjectChartDelete(0,InpName);
   ChartRedraw();
//--- attente d'1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_BITMAP

Objet Bitmap.



Note

Pour l'objet Bitmap, vous pouvez sélectionner la zone visible d'une image.

Exemple

Le script suivant crée plusieurs bitmaps sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée un bitmap dans la fenêtre du graphique."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpFile="\\Images\\dollar.bmp"; // Nom du fichier Bitmap
input int             InpWidth=24;                    // Coordonnée X de la zone visib
input int             InpHeight=24;                   // Coordonnée Y de la zone visib
input int             InpXOffset=4;                   // Décalage de la zone visible s
input int             InpYOffset=4;                   // Décalage de la zone visible s
input color           InpColor=clrRed;                // Couleur de la bordure au mome
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;           // Style de ligne au moment de l
input int             InpPointWidth=1;                // Taille du point à déplacer
input bool            InpBack=false;                  // Objet en arrière-plan
input bool            InpSelection=false;             // Mise en surbrillance pour dép
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpHidden=true;                 // Cahcé dans la liste des objet
input long            InpZOrder=0;                    // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée une bitmap dans la fenêtre du graphique
//+------------------------------------------------------------------------------------
bool BitmapCreate(const long            chart_ID=0,        // identifiant du graphique
                  const string          name="Bitmap",     // nom du bitmap
                  const int             sub_window=0,      // indice de sous-fenêtre
                  datetime              time=0,            // heure du point d'ancrage
                  double                price=0,           // prix du point d'ancrage
                  const string          file="",           // nom du fichier bitmap
                  const int             width=10,          // Coordonnée X de la zone
                  const int             height=10,         // Coordonnée Y de la zone
                  const int             x_offset=0,        // Décalage de la zone visi
                  const int             y_offset=0,        // Décalage de la zone visi
                  const color           clr=clrRed,        // couleur de la bordure en
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // style de trait en surbri
                  const int             point_width=1,     // taille du point de dépla
                  const bool            back=false,        // en arrière-plan
                  const bool            selection=false,   // mise en surbrillance pou
                  const bool            hidden=true,       // caché dans la liste des
                  const long            z_order=0)         // priorité du clic de sour
  {
//--- Définit les coordonnées du point d'ancrage si elles ne sont pas définies
   ChangeBitmapEmptyPoint(time,price);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée une bitmap
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer une bitmap dans la fenêtre du graphique ! Code
      return(false);
     }
//--- définit le chemin d'accès au fichier image
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            " : impossible de charger l'image ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- définit la zone visible de l'omage ; si la largeur et la hauteur
//--- excèdent la largeur et la hauteur (respectivement) d'une image source,
//--- elle n'est pas dessinée ; dans le cas opposé,
//--- seule la partie correspondant à ces valeurs sera dessinée
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- définit la partie d'une image à afficher dans la zone visible
//--- la partie par défaut est la partie supérieure gauche de l'image ; les valeurs pe
```

```
//--- d'effectuer un décalage de cette zone pour afficher une autre partie de l'image
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- définit la couleur de la bordure lorsque le mode de mise en surbrillance de l'ob
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure lorsque le mode de mise en surbrillance
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille du point d'ancrage pour déplacer un objet
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de l'étiquette par la
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Définit une nouvelle image pour le bitmap
//+------------------------------------------------------------------------------------
bool BitmapSetImage(const long   chart_ID=0,    // identifiant du graphique
                    const string name="Bitmap", // nom du bitmap
                    const string file="")       // chemin vers le fichier
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit le chemin d'accès au fichier image
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            " : impossible de charger l'image ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace une image bitmap dans la fenêtre du graphique
//+------------------------------------------------------------------------------------
bool BitmapMove(const long   chart_ID=0,    // identifiant du graphique
                const string name="Bitmap", // nom du bitmap
                datetime     time=0,        // heure du point d'ancrage
                double       price=0)       // prix du point d'ancrage
  {
//--- si la position du point n'est pas définie, le déplace vers la barre courante du
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace le point d'ancrage
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer le point d'ancrage ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change la taille de la zone visible (bitmap)
//+------------------------------------------------------------------------------------
bool BitmapChangeSize(const long   chart_ID=0,    // identifiant du graphique
                      const string name="Bitmap", // nom du bitmap
                      const int    width=0,       // largeur du bitmap
                      const int    height=0)      // hauteur du bitmap
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change la taille du bitmap
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            " : impossible de modifier la largeur du bitmap ! Code d'erreur = ",GetLas
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": n'a pas réussi à modifier la hauteur du bitmap ! Code d'erreur = ",GetL
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Change les coordonnées du coin supérieur gauche de la zone visible                 |
//+------------------------------------------------------------------------------------+
bool BitmapMoveVisibleArea(const long   chart_ID=0,    // identifiant du graphique
                           const string name="Bitmap", // nom du bitmap
                           const int    x_offset=0,    // Coordonnée X de la zone visi
                           const int    y_offset=0)    // Coordonnée Y de la zone visi
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change les coordonnées de la zone visible de l'image
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            " : échec du changement de la coordonnée X de la zone visible ! Code d'er
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            " : échec du changement de la coordonnée Y de la zone visible ! Code d'er
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime un bitmap
//+------------------------------------------------------------------------------------
bool BitmapDelete(const long   chart_ID=0,    // identifiant du graphique
                  const string name="Bitmap") // nom du bitmap
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'étiquette
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à supprimer le bitmap ! Code d'erreur = ",GetLastError
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Vérifie les valeurs du point d'ancrage et définit les valeurs par défaut
//| pour les points vides
//+------------------------------------------------------------------------------------
void ChangeBitmapEmptyPoint(datetime &time,double &price)
  {
//--- si l'heure du point n'est pas définie, elle sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas défini, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------
void OnStart()
  {
   datetime date[];  // tableau pour stocker les dates des barres visibles
   double   close[]; // tableau pour le stockage des prix de Clôture
//--- nom du fichier bitmap
   string   file="\\Images\\dollar.bmp";
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- allocation mémoire
   ArrayResize(date,bars);
   ArrayResize(close,bars);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- remplit le tableau des prix de Clôture
   if(CopyClose(Symbol(),Period(),0,bars,close)==-1)
     {
      Print("Impossible de copier les valeurs des prix de Clôture ! Code d'erreur = ",
      return;
     }
//--- définit à quelle fréquence les images doivent être affichées
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- définit le pas
   int step=1;
   switch(scale)
     {
      case 0:
         step=27;
         break;
      case 1:
         step=14;
         break;
      case 2:
         step=7;
         break;
      case 3:
         step=4;
         break;
      case 4:
         step=2;
         break;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- crée les bitmaps pour les valeurs High et Low des barres (avec les gaps)
   for(int i=0;i<bars;i+=step)
     {
      //--- crée les bitmaps
      if(!BitmapCreate(0,"Bitmap_"+(string)i,0,date[i],close[i],InpFile,InpWidth,InpHe
         InpYOffset,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
        {
         return;
        }
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- supprime les signaux de Vente
   for(int i=0;i<bars;i+=step)
     {
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_BITMAP_LABEL

Objet étiquette Bitmap.

Note

La position du point d'ancrage par rapport à l'étiquette peut être sélectionnée dans l'énumération ENUM_ANCHOR_POINT. Les coordonnées du point d'ancrage sont définies en pixels.

Vous pouvez également sélectionner le coin d'ancrage du bitmap dans l'énumération ENUM_BASE_CORNER.

Pour l'étiquette bitmap, vous pouvez sélectionner la zone visible d'une image.

Exemple

Le script suivant crée plusieurs bitmaps sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée un objet \"Etiquette Bitmap\" objet."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string            InpName="BmpLabel";                // Nom de l'étiquette
input string            InpFileOn="\\Images\\dollar.bmp"; // Nom du fichier en mode Or
input string            InpFileOff="\\Images\\euro.bmp";  // Nom du fichier en mode Of
input bool              InpState=false;                    // Etiquette Appuyée/Relâché
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER;       // Coin d'ancrage dans le gr
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER;           // Type d'ancrage
```

© 2000-2024, MetaQuotes Ltd.

```
input color             InpColor=clrRed;                  // Couleur de la bordure en
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;             // Style de trait en surbri
input int               InpPointWidth=1;                  // Taille du point à déplace
input bool              InpBack=false;                    // Objet en arrière-plan
input bool              InpSelection=false;               // Mise en surbrillance pou
input bool              InpHidden=true;                   // Caché dans la liste des
input long              InpZOrder=0;                      // Priorité du clic de sour
//+------------------------------------------------------------------------------------
//| Crée l'objet de l'Etiquette Bitmap
//+------------------------------------------------------------------------------------
bool BitmapLabelCreate(const long              chart_ID=0,               // identifiar
                       const string            name="BmpLabel",          // nom de l'é
                       const int               sub_window=0,             // indice de
                       const int               x=0,                      // coordonnée
                       const int               y=0,                      // coordonnée
                       const string            file_on="",               // image en r
                       const string            file_off="",              // image en r
                       const int               width=0,                  // Coordonnée
                       const int               height=0,                 // Coordonnée
                       const int               x_offset=10,              // Décalage
                       const int               y_offset=10,              // Décalage
                       const bool              state=false,              // appuyé/rel
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // coin du gr
                       const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // type d'anc
                       const color             clr=clrRed,               // couleur de
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // Style de t
                       const int               point_width=1,            // déplace la
                       const bool              back=false,               // en arrière
                       const bool              selection=false,          // mise en su
                       const bool              hidden=true,              // caché dans
                       const long              z_order=0)                // priorité
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée une étiquette bitmap
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            " : n'a pas réussi à créer l'objet \"Etiquette Bitmap\" ! Code d'erreur =
      return(false);
     }
//--- définit les images en mode On et Off
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,0,file_on))
     {
      Print(__FUNCTION__,
            ": impossible de charger l'image en mode On ! Code d'erreur = ",GetLastEr
      return(false);
     }
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,1,file_off))
```

```
      {
       Print(__FUNCTION__,
             " : impossible de charger l'image en mode Off ! Code d'erreur = ",GetLastE
       return(false);
      }
//--- définit les coordonnées de l'étiquette
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- définit la zone visible de l'omage ; si la largeur et la hauteur
//--- excèdent la largeur et la hauteur (respectivement) d'une image source,
//--- elle n'est pas dessinée ; dans le cas opposé,
//--- seule la partie correspondant à ces valeurs sera dessinée
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- définit la partie d'une image à afficher dans la zone visible
//--- la partie par défaut est la partie supérieure gauche de l'image ; les valeurs pe
//--- d'effectuer un décalage de cette zone pour afficher une autre partie de l'image
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- définit le statut de l'étiquette (appuyé ou relâché)
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- définit le coin du graphique, relativement auquel les coordonnées du point sont
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- définit le type d'ancrage
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- définit la couleur de la bordure lorsque le mode de mise en surbrillance de l'ob
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure lorsque le mode de mise en surbrillance
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la taille du point d'ancrage pour déplacer un objet
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de l'étiquette par la
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Définit une nouvelle image pour l'étiquette de l'objet Bitmap
//+------------------------------------------------------------------------------------
bool BitmapLabelSetImage(const long   chart_ID=0,       // identifiant du graphique
                         const string name="BmpLabel", // nom de l'étiquette
                         const int    on_off=0,         // modificateur (On ou Off)
```

```
                          const string file="")         // chemin vers le fichier
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- définit le chemin d'accès au fichier image
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,on_off,file))
     {
      Print(__FUNCTION__,
            " : impossible de charger l'image ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace l'objet de l'Etiquette Bitmap
//+------------------------------------------------------------------------------------
bool BitmapLabelMove(const long   chart_ID=0,      // identifiant du graphique
                     const string name="BmpLabel", // nom de l'étiquette
                     const int    x=0,             // coordonnée X
                     const int    y=0)             // coordonnée y
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la coordonnée X de l'objet ! Code d'erreur = ",
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la coordonnée Y de l'objet ! Code d'erreur = ",
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change la taille de la zone visible (objet)
//+------------------------------------------------------------------------------------
bool BitmapLabelChangeSize(const long   chart_ID=0,      // identifiant du graphique
                           const string name="BmpLabel", // nom de l'étiquette
                           const int    width=0,         // largeur de l'étiquette
                           const int    height=0)        // hauteur de l'étiquette
  {
//--- réinitialise la valeur d'erreur
```

© 2000-2024, MetaQuotes Ltd.

```
   ResetLastError();
//--- modifie la taille de l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            " : impossible de modifier la largeur de l'objet ! Code d'erreur = ",GetLa
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            " : impossible de modifier la heuteur de l'objet ! Code d'erreur = ",GetLa
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Change les coordonnées du coin supérieur gauche de la zone visible           |
//+------------------------------------------------------------------------------+
bool BitmapLabelMoveVisibleArea(const long   chart_ID=0,      // identifiant du graphi
                                const string name="BmpLabel", // nom de l'étiquette
                                const int    x_offset=0,      // Coordonnée X de la zo
                                const int    y_offset=0)      // Coordonnée Y de la zo
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- change les coordonnées de l'objet de la zone visible
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            " : échec du changement de la coordonnée X de la zone visible ! Code d'err
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            " : échec du changement de la coordonnée Y de la zone visible ! Code d'err
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------
//| Supprime l'objet "Etiquette Bitmap"
//+------------------------------------------------------------------------------
bool BitmapLabelDelete(const long   chart_ID=0,      // identifiant du graphique
                       const string name="BmpLabel") // nom de l'étiquette
  {
```

```
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'étiquette
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : impossible de supprimer l'objet \"Etiquette Bitmap\" ! Code d'erreur =
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- taille de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définit la taille de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Impossible d'obtenir la largeur du graphique ! Code d'erreur = ",GetLastE
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Impossible d'obtenir la hauteur du graphique ! Code d'erreur = ",GetLastE
      return;
     }
//--- définit les coordonnées de l'étiquette bitmap
   int x=(int)x_distance/2;
   int y=(int)y_distance/2;
//--- définit la taille de l'étiquette et les coordonnées de la zone visible
   int width=32;
   int height=32;
   int x_offset=0;
   int y_offset=0;
//--- place l'étiquette bitmap au centre de la fenêtre
   if(!BitmapLabelCreate(0,InpName,0,x,y,InpFileOn,InpFileOff,width,height,x_offset,y_
      InpCorner,InpAnchor,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidd
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- change la taille de la zone visible de l'étiquette dans la boucle
```

© 2000-2024, MetaQuotes Ltd.

```
   for(int i=0;i<6;i++)
     {
      //--- change la taille de la zone visible
      width--;
      height--;
      if(!BitmapLabelChangeSize(0,InpName,width,height))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // 0.3 seconde de délai
      Sleep(300);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- change les coordonnées de la zone visible de l'étiquette dans la boucle
   for(int i=0;i<2;i++)
     {
      //--- change les coordonnées de la zone visible
      x_offset++;
      y_offset++;
      if(!BitmapLabelMoveVisibleArea(0,InpName,x_offset,y_offset))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // 0.3 seconde de délai
      Sleep(300);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime l'étiquette
   BitmapLabelDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_EDIT

Objet Zone d'Edition.

Note

Les coordonnées du point d'ancrage sont définies en pixels. Vous pouvez sélectionner le coin d'ancrage de la Zone d'Edition avec l'énumération ENUM_BASE_CORNER.

Vous pouvez aussi sélectionner l'un des types d'alignement du texte dans la Zone d'Edition avec l'énumération ENUM_ALIGN_MODE.

Exemple

Le script suivant crée et déplace l'objet Zone d'Edition sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script crée un objet \"Zone d'Edition\" object."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string           InpName="Edit";              // nom de l'objet
input string           InpText="Text";              // Texte de l'objet
input string           InpFont="Arial";             // Police de caractères
input int              InpFontSize=14;              // Taille de la police de caractèr
input ENUM_ALIGN_MODE  InpAlign=ALIGN_CENTER;       // Type d'alignement du texte
input bool             InpReadOnly=false;           // Autorisation d'édition
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Coin du graphique pour l'ancrag
input color            InpColor=clrBlack;           // Couleur du texte
```

© 2000-2024, MetaQuotes Ltd.

```
input color            InpBackColor=clrWhite;       // Couleur d'arrière plan
input color            InpBorderColor=clrBlack;     // Couleur de la bordure
input bool             InpBack=false;               // Objet en arrière-plan
input bool             InpSelection=false;          // Mise en surbrillance pour dépla
input bool             InpHidden=true;              // Caché dans la liste des objets
input long             InpZOrder=0;                 // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Crée l'objet Zone d'Edition
//+------------------------------------------------------------------------------------
bool EditCreate(const long             chart_ID=0,                // identifiant du gra
                const string           name="Edit",               // nom de l'objet
                const int              sub_window=0,              // indice de sous-fen
                const int              x=0,                       // Coordonnée X
                const int              y=0,                       // Coordonnée Y
                const int              width=50,                  // largeur
                const int              height=18,                 // hauteur
                const string           text="Text",               // texte
                const string           font="Arial",              // police de caractèr
                const int              font_size=10,              // taille de la polic
                const ENUM_ALIGN_MODE  align=ALIGN_CENTER,        // type d'alignement
                const bool             read_only=false,           // autoristation d'éc
                const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER,  // coin du graphique
                const color            clr=clrBlack,              // couleur du texte
                const color            back_clr=clrWhite,         // Couleur d'arrière
                const color            border_clr=clrNONE,        // Couleur de la borc
                const bool             back=false,                // en arrière plan
                const bool             selection=false,           // mise en surbrillar
                const bool             hidden=true,               // caché dans la list
                const long             z_order=0)                 // priorité pour le c
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée le champ d'édition
   if(!ObjectCreate(chart_ID,name,OBJ_EDIT,sub_window,0,0))
     {
      Print(__FUNCTION__,
            " : échec de création de l'objet \"Zone d'Edition\" object! Code d'erreur
      return(false);
     }
//--- définit les coordonnées de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- définit la taille de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- définit le texte
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- définit la police de caractères du texte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit la taille de la police de caractères
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- définit le type d'alignement du texte dans l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_ALIGN,align);
//--- active (true) ou annule (false) le mode lecture-seule
   ObjectSetInteger(chart_ID,name,OBJPROP_READONLY,read_only);
//--- définit le coin du graphique à partir duquel les coordonnées de l'objet sont dé
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- définit la couleur du texte
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit la couleur d'arrière-plan
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- définit la couleur de la bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de l'étiquette par la
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace l'objet Zone d'Edition
//+------------------------------------------------------------------------------------
bool EditMove(const long   chart_ID=0,  // identifiant du graphique
              const string name="Edit", // nom de l'objet
              const int    x=0,         // Coordonnée X
              const int    y=0)         // Coordonnée Y
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la coordonnée X de l'objet ! Code d'erreur = ",
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la coordonnée Y de l'objet ! Code d'erreur = ",
      return(false);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Redimmensionne l'objet Zone d'Edition
//+------------------------------------------------------------------------------------
bool EditChangeSize(const long   chart_ID=0,  // identifiant du graphique
                    const string name="Edit", // nom de l'objet
                    const int    width=0,     // largeur
                    const int    height=0)    // hauteur
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie la taille de l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            " : impossible de modifier la largeur de l'objet ! Code d'erreur = ",GetLa
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            " : impossible de modifier la heuteur de l'objet ! Code d'erreur = ",GetLa
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change le texte de l'objet Zone d'Edition
//+------------------------------------------------------------------------------------
bool EditTextChange(const long   chart_ID=0,  // identifiant du graphique
                    const string name="Edit", // nom de l'objet
                    const string text="Text") // texte
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le texte de l'objet
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le texte ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
//| Retourne le texte de l'objet Zone d'Edition
//+------------------------------------------------------------------------------
bool EditTextGet(string      &text,        // texte
                 const long   chart_ID=0,  // identifiant du graphique
                 const string name="Edit") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- récupère le texte de l'objet
   if(!ObjectGetString(chart_ID,name,OBJPROP_TEXT,0,text))
     {
      Print(__FUNCTION__,
            " : échec de récupération du texte ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------
//| Supprime l'objet Zone d'Edition
//+------------------------------------------------------------------------------
bool EditDelete(const long   chart_ID=0,  // identifiant du graphique
                const string name="Edit") // nom de l'objet
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'étiquette
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            " : échec de la suppression de l'objet \"Zone d'Edition\" ! Code d'erreur
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------
void OnStart()
  {
//--- taille de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définit la taille de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Impossible d'obtenir la largeur du graphique ! Code d'erreur = ",GetLastE
      return;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
      {
       Print("Impossible d'obtenir la hauteur du graphique ! Code d'erreur = ",GetLastE
       return;
      }
//--- définit le pas de changement du champ d'édition
   int x_step=(int)x_distance/64;
//--- définit les coordonnées et la taille du champ d'édition
   int x=(int)x_distance/8;
   int y=(int)y_distance/2;
   int x_size=(int)x_distance/8;
   int y_size=InpFontSize*2;
//--- stocke le texte dans la variable locale
   string text=InpText;
//--- crée le champ d'édition
   if(!EditCreate(0,InpName,0,x,y,x_size,y_size,InpText,InpFont,InpFontSize,InpAlign,I
       InpCorner,InpColor,InpBackColor,InpBorderColor,InpBack,InpSelection,InpHidden,In
      {
       return;
      }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- étend le champ d'édition
   while(x_size-x<x_distance*5/8)
      {
       //--- augmente la largeur du champ d'édition
       x_size+=x_step;
       if(!EditChangeSize(0,InpName,x_size,y_size))
          return;
       //--- vérifie si l'exécution du script a été stoppée
       if(IsStopped())
          return;
       //--- redessine le graphique et attend 0.05 seconde
       ChartRedraw();
       Sleep(50);
      }
//--- délai d'une demie seconde
   Sleep(500);
//--- change le texte
   for(int i=0;i<20;i++)
      {
       //--- ajoute "+" au début et à la fin
       text="+"+text+"+";
       if(!EditTextChange(0,InpName,text))
          return;
       //--- vérifie si l'exécution du script a été stoppée
       if(IsStopped())
```

© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- redessine le graphique et attend 0.1 seconde
      ChartRedraw();
      Sleep(100);
     }
//--- délai d'une demie seconde
   Sleep(500);
//--- supprime le champ d'édition
   EditDelete(0,InpName);
   ChartRedraw();
//--- attente d'1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_EVENT

Objet évènement.

Note

Lorsque la souris passe au-dessus de l'évènement, son texte apparaît.

Exemple

Le script suivant crée et déplace l'objet Evènement sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un objet graphique \"Evènement\"."
#property description "Le point d'ancrage de la date est défini en pourcentage de"
#property description "la largeur de la fenêtre du graphique en barres."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string          InpName="Event";     // Nom de l'évènement
input int             InpDate=25;          // Date de l'évènement, %
input string          InpText="Text";      // Texte de l'évènement
input color           InpColor=clrRed;     // Couleur de l'évènement
input int             InpWidth=1;          // Taille du point pour le déplacement
input bool            InpBack=false;       // Evènement en arrière plan
input bool            InpSelection=false;  // mise en surbrillance pour le déplacement
input bool            InpHidden=true;      // caché dans la liste des objets
```

© 2000-2024, MetaQuotes Ltd.

```
input long            InpZOrder=0;        // priorité pour le clic de souris
//+------------------------------------------------------------------------------
//| Crée l'objet Evènement sur le graphique
//+------------------------------------------------------------------------------
bool EventCreate(const long            chart_ID=0,       // identifiant du graphique
                 const string          name="Event",     // nom de l'évènement
                 const int             sub_window=0,     // indice de sous-fenêtre
                 const string          text="Text",      // texte de l'évènement
                 datetime              time=0,           // heure
                 const color           clr=clrRed,       // couleur
                 const int             width=1,          // largeur du point lors de la
                 const bool            back=false,       // en arrière plan
                 const bool            selection=false,  // mise en surbrillance pour l
                 const bool            hidden=true,      // caché dans la liste des obj
                 const long            z_order=0)        // priorité pour le clic de sc
  {
//--- si l'heure n'est pas définie, crée l'objet sur la dernière barre
   if(!time)
      time=TimeCurrent();
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée l'objet Evènement
   if(!ObjectCreate(chart_ID,name,OBJ_EVENT,sub_window,time,0))
     {
      Print(__FUNCTION__,
            " : échec de création de l'objet \"Evènement\" ! Code d'erreur = ",GetLast
      return(false);
     }
//--- définit le texte de l'évènement
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- définit la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit la largeur du point d'ancrage si l'objet est mis en surbrillance
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de l'évènement à la sc
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des c
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------
//| Change le texte de l'objet Evènement
//+------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
bool EventTextChange(const long   chart_ID=0,   // identifiant du graphique
                     const string name="Event", // nom de l'évènement
                     const string text="Text")  // texte
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le texte de l'objet
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le texte ! Code d'erreur = ",GetLastError());
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace l'objet Evènement
//+------------------------------------------------------------------------------------
bool EventMove(const long   chart_ID=0,   // identifiant du graphique
               const string name="Event", // nom de l'évènement
               datetime     time=0)       // heure
  {
//--- si l'heure n'est pas définie, déplace l'évènement sur la dernière barre
   if(!time)
      time=TimeCurrent();
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace l'objet
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            " : échec du déplacement de l'objet \"Evènement\" ! Code d'erreur = ",GetI
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime l'objet Evènement
//+------------------------------------------------------------------------------------
bool EventDelete(const long   chart_ID=0,   // identifiant du graphique
                 const string name="Event") // nom de l'évènement
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'objet
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            " : échec de suppression de l'objet \"Evènement\" ! Code d'erreur = ",GetI
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------
void OnStart()
  {
//--- vérifie l'exactitude des paramètres d'entrée
   if(InpDate<0 || InpDate>100)
     {
      Print("Erreur ! Valeurs incorrectes des paramètres d'entrée !");
      return;
     }
//--- nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tableau pour stocker les valeurs de date à utiliser
//--- pour définir et changer les coordonnées des points d'ancrage de l'objet Evènemen
   datetime date[];
//--- allocation mémoire
   ArrayResize(date,bars);
//--- remplit le tableau de dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("N'a pas réussi à copier les valeurs de l'heure ! Code d'erreur = ",GetLas
      return;
     }
//--- définit les points pour créer un objet
   int d=InpDate*(bars-1)/100;
//--- crée l'objet Evènement
   if(!EventCreate(0,InpName,0,InpText,date[d],InpColor,InpWidth,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- déplace maintenant l'objet
//--- compteur de boucle
   int h_steps=bars/2;
//--- déplace l'objet
   for(int i=0;i<h_steps;i++)
     {
```

```
      //--- utilise la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- déplace le point
      if(!EventMove(0,InpName,date[d]))
         return;
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique
      ChartRedraw();
      // délai de 0,05 secondes
      Sleep(50);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- supprime le canal du graphique
   EventDelete(0,InpName);
   ChartRedraw();
//--- délai de 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_RECTANGLE_LABEL

Objet Etiquette Rectangle.

Note

Les coordonnées du point d'ancrage sont définies en pixels. Vous pouvez sélectionner le coin d'ancrage de l'étiquette rectangle dans l'énumération ENUM_BASE_CORNER. Le type de bordure de l'étiquette rectangle peut être sélectionnée dans l'énumération ENUM_BORDER_TYPE.

L'objet est utilisé pour créer et concevoir une interface graphique personnalisée.

Exemple

Le script suivant crée et déplace un objet Etiquette Rectangle sur le graphique. Des fonctions spéciales ont été développées pour créer et modifier les propriétés de l'objet graphique. Vous pouvez utiliser ces fonctions "telles quelles" dans vos propres applications.

```
//--- description
#property description "Le script dessine un objet graphique \"Etiquette Rectangle\"."
//--- affichage de la fenêtre des paramètres d'entrée lors du lancement du script
#property script_show_inputs
//--- paramètres d'entrée du script
input string           InpName="RectLabel";          // Nom de l'étiquette
input color            InpBackColor=clrSkyBlue;      // Couleur de l'arrière-plan
input ENUM_BORDER_TYPE InpBorder=BORDER_FLAT;        // Type de bordure
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Coin du graphique pour l'ancrag
input color            InpColor=clrDarkBlue;         // Couleur de la bordure plate (Fl
input ENUM_LINE_STYLE  InpStyle=STYLE_SOLID;         // style de bordure plat (Flat)
input int              InpLineWidth=3;               // largeur de la bordure plate (Fl
```

© 2000-2024, MetaQuotes Ltd.

```
input bool              InpBack=false;                // Objet en arrière-plan
input bool              InpSelection=true;            // Mise en surbrillance pour dépla
input bool              InpHidden=true;               // Caché dans la liste des objets
input long              InpZOrder=0;                  // Priorité du clic de souris
//+------------------------------------------------------------------------------------
//| Créer une étiquette rectangle
//+------------------------------------------------------------------------------------
bool RectLabelCreate(const long             chart_ID=0,                // identifiant d
                     const string           name="RectLabel",          // nom de l'étic
                     const int              sub_window=0,              // indice de sou
                     const int              x=0,                       // Coordonnée X
                     const int              y=0,                       // Coordonnée Y
                     const int              width=50,                  // largeur
                     const int              height=18,                 // hauteur
                     const color            back_clr=C'236,233,216',   // couleur en ar
                     const ENUM_BORDER_TYPE border=BORDER_SUNKEN,      // type de bordu
                     const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER,  // coin du graph
                     const color            clr=clrRed,                // Couleur de la
                     const ENUM_LINE_STYLE  style=STYLE_SOLID,         // style de la b
                     const int              line_width=1,              // largeur de la
                     const bool             back=false,                // en arrière-pl
                     const bool             selection=false,           // mise en surbr
                     const bool             hidden=true,               // caché dans la
                     const long             z_order=0)                 // priorité du d
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- crée l'étiquette rectangle
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            " : impossible de créer l'étiquette rectangle ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- définit les coordonnées de l'étiquette
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- définit la taille de l'étiquette
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- définit la couleur d'arrière-plan
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- définit le type de bordure
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border);
//--- définit le coin du graphique, relativement auquel les coordonnées du point sont
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- définit la couleur de la bordure plate (dans le mode Plat)
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- définit le style de ligne de la bordure plate
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- définit la largeur de la bordure plate
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,line_width);
//--- affiche en premier plan (false) ou en arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- active (true) ou désactive (false) le mode de déplacement de l'étiquette par la
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- masque (true) ou affiche (false) le nom de l'objet graphique dans la liste des c
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- définit la priorité pour recevoir l'événement d'un clic de souris dans le graphi
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Déplace l'étiquette rectangle
//+------------------------------------------------------------------------------------
bool RectLabelMove(const long   chart_ID=0,       // identifiant du graphique
                   const string name="RectLabel", // nom de l'étiquette
                   const int    x=0,              // coordonnée X
                   const int    y=0)              // coordonnée Y
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- déplace l'étiquette rectangle
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la coordonnée X de l'étiquette ! Code d'erreur
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            " : impossible de déplacer la coordonnée Y de l'étiquette ! Code d'erreur
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Modifie la taille de l'étiquette rectangle
//+------------------------------------------------------------------------------------
bool RectLabelChangeSize(const long   chart_ID=0,       // identifiant du graphique
                         const string name="RectLabel", // nom de l'étiquette
                         const int    width=50,         // largeur de l'étiquette
                         const int    height=18)        // hauteur de l'étiquette
  {
```

```
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie la taille de l'étiquette
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            " : impossible de modifier la largeur de l'étiquette ! Code d'erreur = ",G
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            " : impossible de modifier la hauteur de l'étiquette ! Code d'erreur = ",G
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Change le type de bordure de l'étiquette rectangle
//+------------------------------------------------------------------------------------
bool RectLabelChangeBorderType(const long             chart_ID=0,           // identif
                               const string           name="RectLabel",     // nom de
                               const ENUM_BORDER_TYPE border=BORDER_SUNKEN) // type de
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- modifie le type de bordure
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border))
     {
      Print(__FUNCTION__,
            " : impossible de modifier le type de bordure ! Code d'erreur = ",GetLastE
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Supprime l'étiquette rectangle
//+------------------------------------------------------------------------------------
bool RectLabelDelete(const long   chart_ID=0,       // identifiant du graphique
                     const string name="RectLabel") // nom de l'étiquette
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- supprime l'étiquette
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

© 2000-2024, MetaQuotes Ltd.

```
            " : impossible de supprimer l'étiquette rectangle ! Code d'erreur = ",GetI
      return(false);
     }
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- taille de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définit la taille de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Impossible d'obtenir la largeur du graphique ! Code d'erreur = ",GetLastE
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Impossible d'obtenir la hauteur du graphique ! Code d'erreur = ",GetLastE
      return;
     }
//--- définit les coordonnées de l'étiquette rectangle
   int x=(int)x_distance/4;
   int y=(int)y_distance/4;
//--- définit la taille de l'étiquette
   int width=(int)x_distance/4;
   int height=(int)y_distance/4;
//--- crée l'étiquette rectangle
   if(!RectLabelCreate(0,InpName,0,x,y,width,height,InpBackColor,InpBorder,InpCorner,
      InpColor,InpStyle,InpLineWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- modifie la taille de l'étiquette rectangle
   int steps=(int)MathMin(x_distance/4,y_distance/4);
   for(int i=0;i<steps;i++)
     {
      //--- redimmensionnement
      width+=1;
      height+=1;
      if(!RectLabelChangeSize(0,InpName,width,height))
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- vérifie si l'exécution du script a été stoppée
      if(IsStopped())
         return;
      //--- redessine le graphique et attend 0.01 seconde
      ChartRedraw();
      Sleep(10);
     }
//--- délai de 1 seconde
   Sleep(1000);
//--- modifie le type de bordure
   if(!RectLabelChangeBorderType(0,InpName,BORDER_RAISED))
      return;
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- modifie le type de bordure
   if(!RectLabelChangeBorderType(0,InpName,BORDER_SUNKEN))
      return;
//--- redéssine le graphique et attend 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- supprime l'étiquette
   RectLabelDelete(0,InpName);
   ChartRedraw();
//--- attente d'1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

Propriété des objets

Les objets graphique peuvent avoir différentes propriétés suivant le type de l'objet. Les valeurs des propriétés de l'objet sont définies et retournées par les fonctions correspondantes permettant de travailler avec les objets graphiques.

Tous les objets utilisés en analyse technique sont liés aux coordonnées de temps et de prix : ligne de tendance, canaux, outils de Fibonacci, etc. Mais il existe aussi des objets auxiliaires permettant d'améliorer l'interface utilisateur et qui sont liés à la partie toujours visible d'un graphique (fenêtre principale du graphique ou sous-fenêtres d'un indicateur) :

Objet	Identifiant	X/Y	Largeur/Hauteur	Date/Prix	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Texte	OBJ_TEXT	—	—	Oui	—	Oui	Oui
Etiquette	OBJ_LABEL	Oui	Oui (en lecture seule)	—	Oui	Oui	Oui
Bouton	OBJ_BUTTON	Oui	Oui	—	Oui	—	—
Bitmap	OBJ_BITMAP	—	Oui (en lecture seule)	Oui	—	Oui	—
Etiquete Bitmap	OBJ_BITMAP_LABEL	Oui	Oui (en lecture seule)	—	Oui	Oui	—
Champ d'Edition	OBJ_EDIT	Oui	Oui	—	Oui	—	—
Etiquette Rectangle	OBJ_RECTANGLE_LABEL	Oui	Oui	—	Oui	—	—

Les considérations suivantes sont utilisées dans le tableau :

- **X/Y** - coordonnées des points d'ancrage spécifiés en pixels relativement à un coin du graphique ;
- **Largeur/Hauteur** - les objets ont une largeur et une hauteur. Pour "lecture seule", la largeur et la hauteur ne sont calculées qu'une seule fois lorsque l'objet est dessiné sur le graphique ;
- **Date/Prix** - les coordonnées du point d'ancrage sont spécifiées avec la date et le prix ;
- **OBJPROP_CORNER** - définit le coin du graphique relativement auquel les coordonnées du point d'ancrage sont spécifiées. Peut être l'une des 4 valeurs de l'énumération ENUM_BASE_CORNER ;
- **OBJPROP_ANCHOR** - définit le point d'ancrage dans l'objet lui-même et peut être l'une des 9 valeurs de l'énumération ENUM_ANCHOR_POINT. Les coordonnées en pixels sont spécifiées depuis ce point particulier vers le coin sélectionné du graphique ;
- **OBJPROP_ANGLE** - définit l'angle de rotation de l'objet dans le sens horaire.

© 2000-2024, MetaQuotes Ltd.

Les fonctions définissant les propriétés des objets graphiques, ainsi que les opérations ObjectCreate() et ObjectMove() pour créer et déplacer des objets sur le graphique, sont en fait utilisés pour envoyer des commandes au graphique. Si ces fonctions sont excutées avec succès, la commande est ajoutée dans la queue générale des évènements du graphique. Les changements visuels des propriétés des objets graphiques sont implémentés lors du traitement de la queue des évènements du graphique.

N'attendez donc pas une mise à jour visuelle immédiate des objets graphiques après un appel à ces fonctions. Généralement, les objets graphiques du graphique sont mis à jour automatiquement par le terminal suivant les évènements de changement - arrivée d'une nouvelle cotation, redimensionnement de la fenêtre du graphique, etc. Utilisez la fonctionChartRedraw() pour forcer la mise à jour des objets graphiques.

Pour les fonctions ObjectSetInteger() et ObjectGetInteger()

ENUM_OBJECT_PROPERTY_INTEGER

Identificateur	Description	Type de la Propriété
OBJPROP_COLOR	Couleur	color
OBJPROP_STYLE	Style	ENUM_LINE_STYLE
OBJPROP_WIDTH	Epaisseur de la ligne	int
OBJPROP_BACK	Objet en arrière-plan	bool
OBJPROP_ZORDER	Priorité de l'objet graphique pou	long

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	r la réception des événements de clic sur un graphique (CHARTEVENT_CLICK). La valeur par défaut zéro est définie lors de la création d'un objet : la pri	

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	orité peut être augmentée si nécessaire. Lorsque les objets sont placés les uns sur les autres, un seul, ayant la plus haute priorité, recevra	

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	l'événement CHARTEVENT_CLICK.	
OBJPROP_FILL	Remplit un objet avec une couleur (pour OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDD	bool

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	EVCHANNEL, OBJ_REGRESSION)	
OBJPROP_HIDDEN	Interdit l'affichage du nom d'un objet graphique dans la liste des objets du menu "Graphiques" - "Objets" - "Liste des	bool

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	objets". La valeur true permet de cacher un objet dans la liste. Par défaut, true est utilisé pour les objets affichant les événements du calendrier	

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	, l'historique de trading et aux objets créés à partir des programmes MQL5. Pour voir ces objets graphiques et accéder à leurs propriétés, cliquez sur	

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	le bouton "Tous" dans la fenêtre "Liste des objets".	
OBJPROP_SELECTED	L'objet est sélectionné	bool
OBJPROP_READONLY	Possibilité d'éditer le texte d'un objet Zone d'Edition	bool
OBJPROP_TYPE	Type d'o	ENUM_OBJECT r/o

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	bjet	
OBJPROP_TIME	Coordonnée en temps	datetime modificateur=numéro du point d'ancrage
OBJPROP_SELECTABLE	Disponibilité de l'objet	bool
OBJPROP_CREATETIME	Création de l'heure de l'objet	datetime r/o
OBJPROP_LEVELS	Nombre de niveaux	int
OBJPROP_LEVELCOLOR	Couleur de ligne de niveau	color modificateur=numéro du niveau
OBJPROP_LEVELSTYLE	Style de	ENUM_LINE_STYLE modificateur=numéro du niveau

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	ligne de niveau	
OBJPROP_LEVELWIDTH	Epaisseur de la ligne de niveau	int modificateur=numéro du niveau
OBJPROP_ALIGN	Alignement horizontal du texte dans l'objet "Zone d'Edition" (OBJ_EDIT)	ENUM_ALIGN_MODE
OBJPROP_FONTSIZE	Taille de police de caract	int

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	ères	
OBJPROP_RAY_LEFT	Prolongation vers la gauche	bool
OBJPROP_RAY_RIGHT	Prolongation vers la droite	bool
OBJPROP_RAY	Une ligne verticale traverse toutes les fenêtres d'un graphique	bool
OBJPROP_ELLIPSE	Affichage de l'ellipsement	bool

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	ière de l'objet Arc de Fibonacci (OBJ_FIBOARC)	
OBJPROP_ARROWCODE	Code de la flèche pour l'objet Flèche	uchar
OBJPROP_TIMEFRAMES	Visibilité de l'objet dans les différentes périodes	ensemble de flags flags
OBJPROP_ANCHOR	Localis	ENUM_ARROW_ANCHOR (pour OBJ_ARROW),

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	ation du point d'ancrage d'un objet graphique	ENUM_ANCHOR_POINT (pour OBJ_LABEL, OBJ_BITMAP_LABEL et OBJ_TEXT)
OBJPROP_XDISTANCE	La distance en pixels suivant l'axe X depuis le coin d'ancrage (voir la note)	int
OBJPROP_YDISTANCE	La distance en pix	int

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	els suivant l'axe Y depuis le coin d'ancrage (voir la note)	
OBJPROP_DIRECTION	Tendance de l'objet de Gann	ENUM_GANN_DIRECTION
OBJPROP_DEGREE	Niveau de la Marque de la Vague d'Elliott	ENUM_ELLIOT_WAVE_DEGREE
OBJPROP_DRAWLINES	Affichage des lign	bool

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	es pour marquer la Vague d'Elliott	
OBJPROP_STATE	Etat du bouton (appuyé / relâché)	bool
OBJPROP_CHART_ID	Identifiant de l'objet "Graphique" (OBJ_CHART). Il permet de travailler ave	long r/o

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	c les propriétés de cet objet comme avec un graphique normal en utilisant les fonctions décrites dans Opérations sur les Graphiques, mais il y a	

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	quelques exceptions.	
OBJPROP_XSIZE	La largeur de l'objet suivant l'axe X en pixels. Spécifié pour les objets OBJ_LABEL (lecture seule), OBJ_BUTTON, OBJ_CHART, OBJ_BI	int

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	TMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_YSIZE	La hauteur de l'objet suivant l'axe Y en pixels. Spécifié pour les objets OBJ_LABEL (lectur	int

© 2000-2024, MetaQuotes Ltd.

Identificateur	Des crip tion	Type de la Propriété
	e seu le), OBJ _BU TT ON, OBJ _C HA RT, OBJ _BI TM AP, OBJ _BI TM AP_ LAB EL, OBJ _ED IT, OBJ _RE CT AN GL E_L ABE L.	
OBJPROP_XOFFSET	La coo rdo nné e X du coi n sup éri eur gau che de la	int

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	zone rectangulaire visible dans les objets graphiques "Etiquette Bitmap" et "Bitmap" (OBJ_BITMAP_LABEL et OBJ_BITMAP). La valeur est définie en	

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	pixels relativement au coin supérieur gauche de l'image d'origine.	
OBJPROP_YOFFSET	La coordonnée Y du coin supérieur gauche de la zone rectangulaire visible dans les	int

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	objets graphiques "Etiquette Bitmap" et "Bitmap" (OBJ_BITMAP_LABEL et OBJ_BITMAP). La valeur est définie en pixels relativement au coin supérieur	

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	gauche de l'image d'origine.	
OBJPROP_PERIOD	Période de l'objet Graphique	ENUM_TIMEFRAMES
OBJPROP_DATE_SCALE	Affichage de l'échelle du temps pour l'objet Graphique	bool
OBJPROP_PRICE_SCALE	Affichage de l'échelle des prix	bool

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	pour l'objet Graphique	
OBJPROP_CHART_SCALE	L'échelle de l'objet Graphique	int valeur dans l'intervalle 0-5
OBJPROP_BGCOLOR	La couleur en arrière-plan pour OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	color
OBJPROP_CORNER	Le coin	ENUM_BASE_CORNER

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	du graphique pour lier un objet graphique	
OBJPROP_BORDER_TYPE	Type de bordure pour l'objet "Etiquette Rectangle"	ENUM_BORDER_TYPE
OBJPROP_BORDER_COLOR	Couleur de bordure des objets OBJ_EDIT et OBJ_BUTTON	color

© 2000-2024, MetaQuotes Ltd.

Lors de l'utilisation des opérations sur les graphiques pour l'objet "Graphique" (OBJ_CHART), les limites suivantes sont imposées :

- Il ne peut pas être fermé avec ChartClose() ;
- Le symbole et la période ne peuvent pas être changés avec la fonction ChartSetSymbolPeriod() ;
- Les propriétés suivantes sont sans effet CHART_SCALE, CHART_BRING_TO_TOP, CHART_SHOW_DATE_SCALE et CHART_SHOW_PRICE_SCALE (ENUM_CHART_PROPERTY_INTEGER).

Vous pouvez définir un mode spécial d'affichage de l'image pour les objets OBJ_BITMAP_LABEL et OBJ_BITMAP . Dans ce mode, seule une partie de l'image d'origine (sur laquelle une zone visible rectangulaire est appliquée) est affichée, tandis que le reste de l'image devient invisible. La taille de cette zone est définie avec les propriétés OBJPROP_XSIZE et OBJPROP_YSIZE. La zone visible ne peut être "déplacée" que dans l'image d'origine en utilisant les propriétés OBJPROP_XOFFSET et OBJPROP_YOFFSET.

Pour les objets de taille fixe : OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT et OBJ_CHART, les propriétés OBJPROP_XDISTANCE et OBJPROP_YDISTANCE définissent la position du point supérieur gauche de l'objet relativement au coin du graphique (OBJPROP_CORNER), à partir duquel les coordonnées X et Y seront comptées en pixels.

Pour les fonctions ObjectSetDouble() et ObjectGetDouble()

ENUM_OBJECT_PROPERTY_DOUBLE

Identificateur	Description	Type de la Propriété
OBJPROP_PRICE	Coordonnée du prix	double modificateur=numéro du point d'ancrage
OBJPROP_LEVELVALUE	Valeur du niveau	double modificateur=numéro du niveau
OBJPROP_SCALE	Echelle (propriét	double

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	és des objets de Gann et des Arcs de Fibonacci)	
OBJPROP_ANGLE	Angle. Pour les objets sans angle spécifié, créé depuis un programme, la valeur est égale à EMPTY	double

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	_VALUE	
OBJPROP_DEVIATION	Déviation pour le Canal de Déviation Standard	double

Pour les fonctions ObjectSetString() et ObjectGetString()

ENUM_OBJECT_PROPERTY_STRING

Identificateur	Description	Type de la Propriété
OBJPROP_NAME	Nom de l'objet	string
OBJPROP_TEXT	Description de l'objet (le texte contenu dans	string

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	l'objet)	
OBJPROP_TOOLTIP	Le texte de l'info bulle. Si la propriété n'est pas définie, alors l'info bulle générée automatiquement par le terminal est affichée. Une info bu	string

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	lle peut être désactivée en lui assignant la valeur "\n" (line feed)	
OBJPROP_LEVELTEXT	Description du niveau	string modificateur=numéro du niveau
OBJPROP_FONT	Police de caractères	string
OBJPROP_BMPFILE	Le nom du fichier BMP pour	string modificateur: 0-état ON, 1-état OFF

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type de la Propriété
	l'Etiquette Bitmap. Voir aussi Ressources	
OBJPROP_SYMBOL	Symbole pour l'objet Graphique	string

Pour l'objet OBJ_RECTANGLE_LABEL ("Etiquette Rectangle"), l'un des trois modes de conception peut être utilisé, auxquels les valeurs suivantes de ENUM_BORDER_TYPE correspondent.

ENUM_BORDER_TYPE

Identificateur	Description
BORDER_FLAT	A plat
BORDER_RAISED	Relevé
BORDER_SUNKEN	Concave

Pour l'objet OBJ_EDIT ("Zone d'Edition") et pour la fonction ChartScreenShot(), vous pouvez spécifier le type d'alignement horizontal en utilisant les valeurs de l'énumération ENUM_ALIGN_MODE.

ENUM_ALIGN_MODE

Identificateur	Description
ALIGN_LEFT	Alignement à gauche

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description
ALIGN_CENTER	Centré (seulement pour l'objet Zone d'Edition)
ALIGN_RIGHT	Alignement à droite

Exemple :

```
#define  UP          "\x0431"
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//---
   string label_name="my_OBJ_LABEL_object";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Objet ",label_name," non trouvé. Code d'erreur = ",GetLastError());
      //--- crée l'objet Etiquette
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- définit la coordonnée X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- définit la coordonnée Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      //--- définit la couleur du texte
      ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
      //--- définit le texte de l'objet Etiquette
      ObjectSetString(0,label_name,OBJPROP_TEXT,UP);
      //--- définit la police de caractères
      ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");
      //--- définit la taille de la police de caractères
      ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);
      //--- 45 degrés de rotation dans le sens horaire
      ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);
      //--- désactive la sélection de la souris
      ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);
      //--- le dessine sur le graphique
      ChartRedraw(0);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Méthode de Lien avec un Objet

Les objets graphiques Texte, Etiquette, Bitmap et Etiquette Bitmap (OBJ_TEXT, OBJ_LABEL, OBJ_BITMAP et OBJ_BITMAP_LABEL) peuvent être ancrés de l'une des 9 façons définies par la propriété OBJPROP_ANCHOR.

Objet	Identifiant	X/Y	Largeur/Hauteur	Date/Prix	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Texte	OBJ_TEXT	—	—	Oui	—	**Oui**	Oui
Etiquette	OBJ_LABEL	Oui	Oui (en lecture seule)	—	Oui	**Oui**	Oui
Bouton	OBJ_BUTTON	Oui	Oui	—	Oui	—	—
Bitmap	OBJ_BITMAP	—	Oui (en lecture seule)	Oui	—	**Oui**	—
Etiquete Bitmap	OBJ_BITMAP_LABEL	Oui	Oui (en lecture seule)	—	Oui	**Oui**	—
Champ d'Edition	OBJ_EDIT	Oui	Oui	—	Oui	—	—
Etiquette Rectangle	OBJ_RECTANGLE_LABEL	Oui	Oui	—	Oui	—	—

Les considérations suivantes sont utilisées dans le tableau :

- **X/Y** - coordonnées des points d'ancrage spécifiés en pixels relativement à un coin du graphique ;
- **Largeur/Hauteur** - les objets ont une largeur et une hauteur. Pour "lecture seule", la largeur et la hauteur ne sont calculées qu'une seule fois lorsque l'objet est dessiné sur le graphique ;
- **Date/Prix** - les coordonnées du point d'ancrage sont spécifiées avec la date et le prix ;
- **OBJPROP_CORNER** - définit le coin du graphique relativement auquel les coordonnées du point d'ancrage sont spécifiées. Peut être l'une des 4 valeurs de l'énumération ENUM_BASE_CORNER ;
- **OBJPROP_ANCHOR** - définit le point d'ancrage dans l'objet lui-même et peut être l'une des 9 valeurs de l'énumération ENUM_ANCHOR_POINT. Les coordonnées en pixels sont spécifiées depuis ce point particulier vers le coin sélectionné du graphique ;
- **OBJPROP_ANGLE** - définit l'angle de rotation de l'objet dans le sens horaire.

La variante nécessaire peut être spécifiée en utilisant la fonction ObjectSetInteger(handle_du_graphique, nom_objet, OBJPROP_ANCHOR, anchor_point_mode), où anchor_point_mode est l'une des valeurs de ENUM_ANCHOR_POINT.

ENUM_ANCHOR_POINT

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description
ANCHOR_LEFT_UPPER	Point d'ancrage dans le coin supérieur gauche
ANCHOR_LEFT	Point d'ancrage à gauche au centre
ANCHOR_LEFT_LOWER	Point d'ancrage dans le coin inférieur gauche
ANCHOR_LOWER	Point d'ancrage en bas au centre
ANCHOR_RIGHT_LOWER	Point d'ancrage dans le coin inférieur droit
ANCHOR_RIGHT	Point d'ancrage à droite au centre
ANCHOR_RIGHT_UPPER	Point d'ancrage dans le coin supérieur droit
ANCHOR_UPPER	Point d'ancrage en haut au centre
ANCHOR_CENTER	Point d'ancrage strictement au centre de l'objet

Les objets OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT et OBJ_CHART ont un point d'ancrage fixe situé dans le coin supérieur gauche (ANCHOR_LEFT_UPPER).

Exemple :

```
string text_name="my_OBJ_TEXT_object";
if(ObjectFind(0,text_name)<0)
  {
   Print("Objet ",text_name," non trouvé. Code d'erreur = ",GetLastError());
   //--- Récupère le prix maximal du graphique
   double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
   //--- Crée l'objet Etiquette
   ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
   //--- Définit la couleur du texteSet color of the text
   ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
   //--- Définit la couleur d'arrière plan
   ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
   //--- Définit le texte pour l'objet Etiquette
   ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
   //--- Définit la police de caractères du texte
   ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
   //--- Définit la taille de la police de caractères
   ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
   //--- Lie l'objet au coin supérieur droit
   ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
   //--- Tourne l'objet de 90 degrés dans le sens horaire
   ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
   //--- Interdit la sélection de l'objet avec la souris
   ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
   //--- redessine l'objet
   ChartRedraw(0);
  }
```

© 2000-2024, MetaQuotes Ltd.

Les objets graphiques Flèche (OBJ_ARROW) ont deux façons de lier leurs coordonnées. Les identifiants sont listés dans ENUM_ARROW_ANCHOR.

ENUM_ARROW_ANCHOR

Identifiant	Description
ANCHOR_TOP	Ancre sur le côté supérieur
ANCHOR_BOTTOM	Ancre sur le côté inférieur

Exemple :

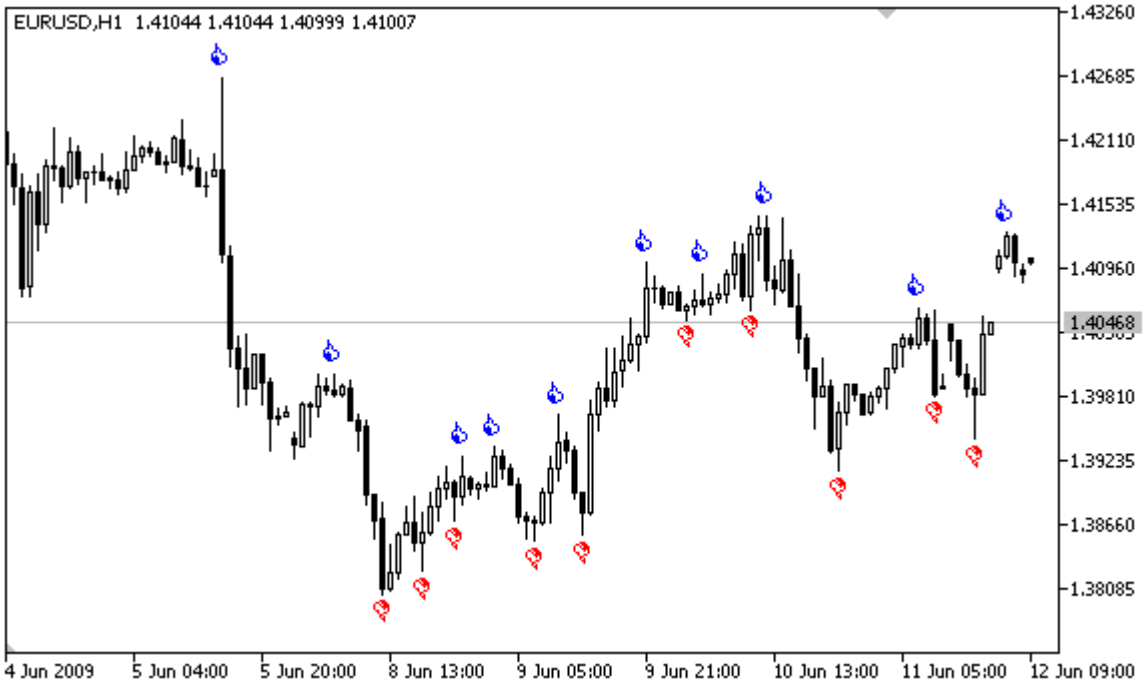
```
void OnStart()
  {
//--- Tableaux supplémentaires
   double Ups[],Downs[];
   datetime Time[];
//--- Définit les tableaux en mode timeseries
   ArraySetAsSeries(Ups,true);
   ArraySetAsSeries(Downs,true);
   ArraySetAsSeries(Time,true);
//--- Crée un handle pour l'indicateur Fractals
   int FractalsHandle=iFractals(NULL,0);
   Print("FractalsHandle = ",FractalsHandle);
//--- Réinitialize la valeur de la dernière erreurSet Last error value to Zero
   ResetLastError();
//--- Essaye de copier les valeurs de l'indicateur
   int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Print("Impossible de cpoier les fractales supérieures. Erreur = ",GetLastError()
      return;
     }

   ResetLastError();
//--- Essaye de copier les valeurs de l'indicateur
   copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
   if(copied<=0)
     {
      Print("Impossible de copier les fractales inférieures. Erreur = ",GetLastError()
      return;
     }

   ResetLastError();
//--- Copies les timeseries contenant les ouvertures des 1000 dernières barres
   copied=CopyTime(NULL,0,0,1000,Time);
   if(copied<=0)
     {
      Print("Impossible de copier l'heure d'Ouverture des 1000 dernières barres");
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }

  int upcounter=0,downcounter=0; // compte du nombre de flèches
  bool created;// reçoit les résultats des tentatives de création d'un objet
  for(int i=2;i<copied;i++)// Boucle sur les valeurs de l'indicateur iFractals
    {
     if(Ups[i]!=EMPTY_VALUE)// Fractale supérieure trouvée
       {
        if(upcounter<10)// Ne crée pas plus de 10 flèches "Vers le haut"
          {
           //--- Essaye de créer un objet "Vers le haut"
           created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[i]
           if(created)// S'il est créé, personnalisons-le
             {
              //--- Le point d'ancrage est en-dessous afin de ne pas recouvrir la bar
              ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
              //--- Touche finale - la couleur
              ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
              upcounter++;
             }
          }
       }
     if(Downs[i]!=EMPTY_VALUE)// Fractale inférieure trouvée
       {
        if(downcounter<10)// Ne crée pas plus de 10 flèches "Vers le bas"
          {
           //--- Essaye de créer un objet "Vers le bas"
           created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Down
           if(created)// S'il est créé, personnalisons-le
             {
              //--- Le point d'ancrage est au-dessus afin de ne pas recouvrir la barr
              ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
              //--- Touche finale - la couleur
              ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
              downcounter++;
             }
          }
       }
    }
}
```

Après l'exécution du script, le graphique ressemblera à celui-ci.

© 2000-2024, MetaQuotes Ltd.



© 2000-2024, MetaQuotes Ltd.

Le Coin du Graphique Auquel un Objet Est Attaché

Il y a un certain nombre d'objets graphiques pour lesquels vous pouvez définir un coin du graphique, relative auquel les coordonnées sont spécifiées en pixels. Ce sont les types d'objets suivants (les identificants des types d'objets sont spécifiés entre parenthèses) :

- Etiquette (OBJ_LABEL) ;
- Bouton (OBJ_BUTTON) ;
- Etiquette Bitmap (OBJ_BITMAP_LABEL) ;
- Champ d'Edition (OBJ_EDIT) ;
- Etiquette Rectangle (OBJ_RECTANGLE_LABEL).

Objet	Identifiant	X/Y	Largeur/Hauteur	Date/Prix	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Texte	OBJ_TEXT	—	—	Oui	—	Oui	Oui
Etiquette	OBJ_LABEL	Oui	Oui (en lecture seule)	—	**Oui**	Oui	Oui
Bouton	OBJ_BUTTON	Oui	Oui	—	**Oui**	—	—
Bitmap	OBJ_BITMAP	—	Oui (en lecture seule)	Oui	—	Oui	—
Etiquete Bitmap	OBJ_BITMAP_LABEL	Oui	Oui (en lecture seule)	—	**Oui**	Oui	—
Champ d'Edition	OBJ_EDIT	Oui	Oui	—	**Oui**	—	—
Etiquette Rectangle	OBJ_RECTANGLE_LABEL	Oui	Oui	—	**Oui**	—	—

Les considérations suivantes sont utilisées dans le tableau :

- **X/Y** - coordonnées des points d'ancrage spécifiés en pixels relativement à un coin du graphique ;
- **Largeur/Hauteur** - les objets ont une largeur et une hauteur. Pour "lecture seule", la largeur et la hauteur ne sont calculées qu'une seule fois lorsque l'objet est dessiné sur le graphique ;
- **Date/Prix** - les coordonnées du point d'ancrage sont spécifiées avec la date et le prix ;
- **OBJPROP_CORNER** - définit le coin du graphique relativement auquel les coordonnées du point d'ancrage sont spécifiées. Peut être l'une des 4 valeurs de l'énumération ENUM_BASE_CORNER ;
- **OBJPROP_ANCHOR** - définit le point d'ancrage dans l'objet lui-même et peut être l'une des 9 valeurs de l'énumération ENUM_ANCHOR_POINT. Les coordonnées en pixels sont spécifiées depuis ce point particulier vers le coin sélectionné du graphique ;

© 2000-2024, MetaQuotes Ltd.

- **OBJPROP_ANGLE** - définit l'angle de rotation de l'objet dans le sens horaire.

Pour spécifier un coin du graphique à partir duquel les coordonnées X et Y sont mesurées en pixels, utilisez ObjectSetInteger(chartID, name, OBJPROP_CORNER, **chart_corner**), où :

- chartID - identifiant du graphique ;
- name - nom d'un objet graphique ;
- OBJPROP_CORNER - identifiant de la propriété pour spécifier le coin d'accrochage ;
- **chart_corner** - le coin du graphique désiré, peut être l'une des valeurs de l'énumération ENUM_BASE_CORNER.

ENUM_BASE_CORNER

Identifiant	Description
CORNER_LEFT_UPPER	Le centre des coordonnées est dans le coin supérieur gauche du graphique
CORNER_LEFT_LOWER	Le centre des coordonnées est dans le coin inférieur gauche du graphique
CORNER_RIGHT_LOWER	Le centre des coordonnées est dans le coin inférieur droit du graphique
CORNER_RIGHT_UPPER	Le centre des coordonnées est dans le coin supérieur droit du graphique

Exemple :

```
void CreateLabel(long   chart_id,
                 string name,
                 int    chart_corner,
                 int    anchor_point,
                 string text_label,
                 int    x_ord,
                 int    y_ord)
  {
//---
   if(ObjectCreate(chart_id,name,OBJ_LABEL,0,0,0))
     {
      ObjectSetInteger(chart_id,name,OBJPROP_CORNER,chart_corner);
      ObjectSetInteger(chart_id,name,OBJPROP_ANCHOR,anchor_point);
      ObjectSetInteger(chart_id,name,OBJPROP_XDISTANCE,x_ord);
      ObjectSetInteger(chart_id,name,OBJPROP_YDISTANCE,y_ord);
      ObjectSetString(chart_id,name,OBJPROP_TEXT,text_label);
     }
   else
      Print("Echec de la créationde l'objet OBJ_LABEL ",name,", Code d'erreur = ", Get
  }
//+------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//---
   int height=(int)ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=(int)ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   string arrows[4]={"LEFT_UPPER","RIGHT_UPPER","RIGHT_LOWER","LEFT_LOWER"};
   CreateLabel(0,arrows[0],CORNER_LEFT_UPPER,ANCHOR_LEFT_UPPER,arrows[0],50,50);
   CreateLabel(0,arrows[1],CORNER_RIGHT_UPPER,ANCHOR_RIGHT_UPPER,arrows[1],50,50);
   CreateLabel(0,arrows[2],CORNER_RIGHT_LOWER,ANCHOR_RIGHT_LOWER,arrows[2],50,50);
   CreateLabel(0,arrows[3],CORNER_LEFT_LOWER,ANCHOR_LEFT_LOWER,arrows[3],50,50);
  }
```

© 2000-2024, MetaQuotes Ltd.

Visibilité des Objets

La conbinaison des flags de visibilité des objets détermine les périodes du graphiques dans lesquelles l'objet est visible. Pour définir/récupérer la valeur de la propriété OBJPROP_TIMEFRAMES, vous pouvez utiliser les fonctions ObjectSetInteger()/ObjectGetInteger().

Identifiant	Valeur	Description
OBJ_NO_PERIODS	0	L'objet n'est dessiné dans aucune période
OBJ_PERIOD_M1	0x00000001	L'objet est dessiné dans le graphique en 1-minute
OBJ_PERIOD_M2	0x00000002	L'objet est dessiné dans le graphique en 2-minutes
OBJ_PERIOD_M3	0x00000004	L'objet est dessiné dans le graphique en 3-minutes
OBJ_PERIOD_M4	0x00000008	L'objet est dessiné dans le graphique en 4-minutes
OBJ_PERIOD_M5	0x00000010	L'objet est dessiné dans le graphique en 5-minutes
OBJ_PERIOD_M6	0x00000020	L'objet est dessiné dans le graphique en 6-minutes
OBJ_PERIOD_M10	0x00000040	L'objet est dessiné dans le graphique en 10-minutes
OBJ_PERIOD_M12	0x00000080	L'objet est dessiné dans le graphique en 12-minutes
OBJ_PERIOD_M15	0x00000100	L'objet est dessiné dans le graphique en 15-minutes
OBJ_PERIOD_M20	0x00000200	L'objet est dessiné dans le graphique en 20-minutes
OBJ_PERIOD_M30	0x00000400	L'objet est dessiné dans le graphique en 30-minutes
OBJ_PERIOD_H1	0x00000800	L'objet est dessiné dans le graphique en 1-heure
OBJ_PERIOD_H2	0x00001000	L'objet est dessiné dans le graphique en 2-heures
OBJ_PERIOD_H3	0x00002000	L'objet est dessiné dans le graphique en 3-heures
OBJ_PERIOD_H4	0x00004000	L'objet est dessiné dans le graphique en 4-heures

© 2000-2024, MetaQuotes Ltd.

Identifiant	Valeur	Description
OBJ_PERIOD_H6	0x00008000	L'objet est dessiné dans le graphique en 6-heures
OBJ_PERIOD_H8	0x00010000	L'objet est dessiné dans le graphique en 8-heures
OBJ_PERIOD_H12	0x00020000	L'objet est dessiné dans le graphique en 12-heures
OBJ_PERIOD_D1	0x00040000	L'objet est dessiné dans le graphique journalier
OBJ_PERIOD_W1	0x00080000	L'objet est dessiné dans le graphique hebdomadaire
OBJ_PERIOD_MN1	0x00100000	L'objet est dessiné dans le graphique mensuel
OBJ_ALL_PERIODS	0x001fffff	L'objet est dessiné sur toutes les périodes

Les flags de visibilité peut être combinés en utilisant le symbole "|", par exemple, la combinaison des flags OBJ_PERIOD_M10|OBJ_PERIOD_H4 signifie que l'objet sera visible dans les périodes 10-minutes et 4-heures.

Exemple :

```
void OnStart()
  {
//---
   string highlevel="PreviousDayHigh";
   string lowlevel="PreviousDayLow";
   double prevHigh;           // Le Plus Haut du jour précédent
   double prevLow;            // Le Plus Bas du jour précédent
   double highs[],lows[];     // Tableaux pour les Plus Haut et les Plus Bas

//---  Réinitialise la dernière erreur
   ResetLastError();
//--- Récupère les 2 derniers Plus Hauts dans la période Daily
   int highsgot=CopyHigh(Symbol(),PERIOD_D1,0,2,highs);
   if(highsgot>0) // Si la copie a échoué
     {
      Print("Les prix les Plus Hauts des 2 derniers jours ont été obtenus avec succès"
      prevHigh=highs[0]; // Le plus haut du jour précédent
      Print("prevHigh = ",prevHigh);
      if(ObjectFind(0,highlevel)<0) // Objet avec le nom highlevel non trouvé
        {
         ObjectCreate(0,highlevel,OBJ_HLINE,0,0,0); // Crée l'objet Ligne Horizontale
        }
      //--- Définit la valeur du niveau de prix pour la ligne highlevel
      ObjectSetDouble(0,highlevel,OBJPROP_PRICE,0,prevHigh);
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- Définit la visibilité sur PERIOD_M10 et PERIOD_H4 uniquement
      ObjectSetInteger(0,highlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else
     {
      Print("Echec de récupération des prix Plus Hauts des 2 derniers jours, Erreur =
     }

//---  Réinitialise la dernière erreur
   ResetLastError();
//--- Récupère les 2 derniers Plus Bas dans la période Daily
   int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2,lows);
   if(lowsgot>0) // Si la copie a réussit
     {
      Print("Les prix les Plus Bas des 2 derniers jours ont été obtenus avec succès");
      prevLow=lows[0]; // Le plus bas du jour précédent
      Print("prevLow = ",prevLow);
      if(ObjectFind(0,lowlevel)<0) // Objet avec le nom lowlevel non trouvé
        {
         ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // Crée l'objet Ligne Horizontale
        }
      //--- Définit la valeur du niveau de prix pour la ligne lowlevel
      ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
      //--- Définit la visibilité sur PERIOD_M10 et PERIOD_H4 uniquement
      ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else Print("Echec de récupération des prix Plus Bas des 2 derniers jours, Erreur =

   ChartRedraw(0); // force le rafraîchissement du graphique
  }
```

Voir aussi

PeriodSeconds, Period, Périodes du Graphique, Date et heure

© 2000-2024, MetaQuotes Ltd.

Niveaux des Vagues d'Elliott

Les Vagues d'Elliott sont représentées avec deux objets graphiques de type OBJ_ELLIOTWAVE5 et OBJ_ELLIOTWAVE3. Pour définir la taille de la vague (méthode d'étiquettage de la vague), la propriété OBJPROP_DEGREE est utilisée, à laquelle l'une des valeurs de l'énumération ENUM_ELLIOT_WAVE_DEGREE peut être assignée.

ENUM_ELLIOT_WAVE_DEGREE

Identifiant	Description
ELLIOTT_GRAND_SUPERCYCLE	Grand Supercycle
ELLIOTT_SUPERCYCLE	Supercycle
ELLIOTT_CYCLE	Cycle
ELLIOTT_PRIMARY	Vague primaire
ELLIOTT_INTERMEDIATE	Vague intermédiaire
ELLIOTT_MINOR	Vague mineure
ELLIOTT_MINUTE	Minute
ELLIOTT_MINUETTE	Minuette
ELLIOTT_SUBMINUETTE	Subminuette

Exemple :

```
for(int i=0;i<ObjectsTotal(0);i++)
  {
   string currobj=ObjectName(0,i);
   if((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE3) ||
      ((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE5)))
     {
      //--- définit le niveau comme INTERMEDIATE
      ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
      //--- affiche des lignes entre les sommets des vagues
      ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
      //--- définit la couleur de ligne
      ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
      //--- définit la largeur de ligne
      ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
      //--- définit la description
      ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Objets de Gann

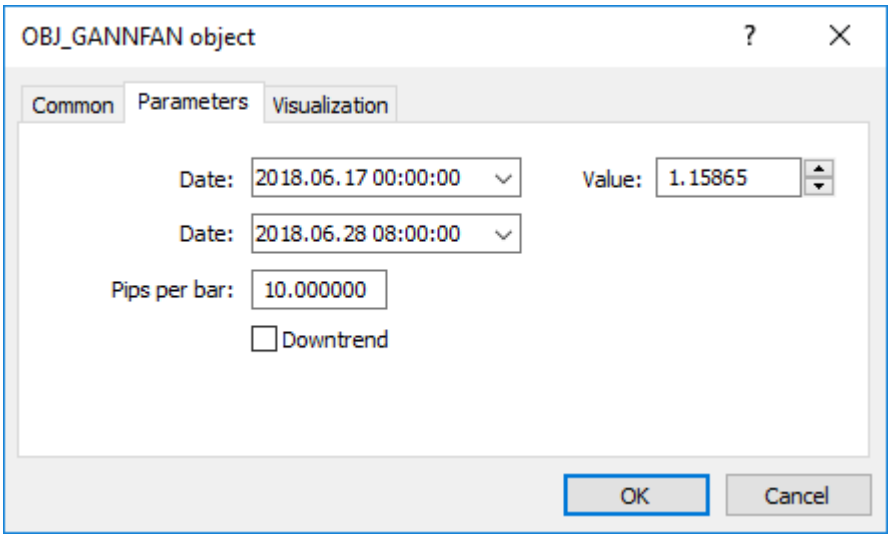
Pour les objets Eventail de Fan (OBJ_GANNFAN) et Grille de Gann (OBJ_GANNGRID), vous pouvez spécifier deux valeurs de l'énumération ENUM_GANN_DIRECTION qui définissent la direction de la tendance.

ENUM_GANN_DIRECTION

Identifiant	Description
GANN_UP_TREND	Ligne correspondant à une ligne de tendance haussière
GANN_DOWN_TREND	Ligne correspondant à une ligne de tendance baissière

Pour définir l'échelle de la ligne principale à 1x1, utilisez la fonction ObjectSetDouble(chart_handle, gann_object_name, OBJPROP_SCALE, scale), où :

- chart_handle - fenêtre du graphique où l'objet est situé ;
- gann_object_name - nom de l'objet ;
- OBJPROP_SCALE - identifiant de la propriété "Echelle" ;
- scale - échelle désirée en Pips/Barre.



Exemple de création d'un Eventail de Gann :

```
void OnStart()
  {
//---
   string my_gann="OBJ_GANNFAN object";
   if(ObjectFind(0,my_gann)<0)// Objet non trouvé
     {
      //--- Information sur l'échec
      Print("Objet ",my_gann," non trouvé. Code d'erreur = ",GetLastError());
      //--- Récupère le prix maximal du graphique
      double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- Récupère le prix minimal du graphique
      double chart_min_price=ChartGetDouble(0,CHART_PRICE_MIN,0);
      //--- Combien de barres sont affichées sur le graphique ?
      int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
      //--- Crée un tableau pour écrire l'heure d'ouverture de chaque barre
      datetime Time[];
      //--- Définit le tableau en tant que timeseries
      ArraySetAsSeries(Time,true);
      //--- Copie maintenant les données des barres visibles dans le graphique dans ce
      int times=CopyTime(NULL,0,0,bars_on_chart,Time);
      if(times<=0)
        {
         Print("Impossible de copier le tableau des heures d'ouverture !");
         return;
        }
      //--- Fin des préparations préliminaires

      //--- Indice de la barre centrale du graphique
      int center_bar=bars_on_chart/2;
      //--- Equateur du graphique - entre le maximum et le minimum
      double mean=(chart_max_price+chart_min_price)/2.0;
      //--- Définit les coordonnées du premier point d'ancrage au centre
      ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
                   //--- Deuxième point d'ancrage à droite
                   Time[center_bar/2],(mean+chart_min_price)/2.0);
      Print("Heure[center_bar] = "+(string)Time[center_bar]+"  Heure[center_bar/2] = "
      //Print("Heure[center_bar]/="+Time[center_bar]+"  HEure[center_bar/2]="+Time[cen
      //--- Définit l'échelle en Pips / Barre
      ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
      //--- Définit la ligne de tendance
      ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
      //--- Définit la largeur de ligne
      ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
      //--- Définit le style de ligne
      ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
      //--- Définit la couleur de ligne
      ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
      //--- Permet à l'utilisateur de sélectionner un objet
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
      //--- Le sélectionne
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
      //--- Le dessine sur le graphique
      ChartRedraw(0);
    }
  }
```

© 2000-2024, MetaQuotes Ltd.

Couleurs Web

Les constantes de couleurs suivantes sont définies pour le type color :

clrBlack	clrDarkGreen	clrDarkSlateGray	clrOlive	clrGreen	clrTeal	clrNavy	clrPurple
clrMaroon	clrIndigo	clrMidnightBlue	clrDarkBlue	clrDarkOliveGreen	clrSaddleBrown	clrForestGreen	clrOliveDrab
clrSeaGreen	clrDarkGoldenrod	clrDarkSlateBlue	clrSienna	clrMediumBlue	clrBrown	clrDarkTurquoise	clrDimGray
clrLightSeaGreen	clrDarkViolet	clrFireBrick	clrMediumVioletRed	clrMediumSeaGreen	clrChocolate	clrCrimson	clrSteelBlue
clrGoldenrod	clrMediumSpringGreen	clrLawnGreen	clrCadetBlue	clrDarkOrchid	clrYellowGreen	clrLimeGreen	clrOrangeRed
clrDarkOrange	clrOrange	clrGold	clrYellow	clrChartreuse	clrLime	clrSpringGreen	clrAqua
clrDeepSkyBlue	clrBlue	clrMagenta	clrRed	clrGray	clrSlateGray	clrPeru	clrBlueViolet
clrLightSlateGray	clrDeepPink	clrMediumTurquoise	clrDodgerBlue	clrTurquoise	clrRoyalBlue	clrSlateBlue	clrDarkKhaki
clrIndianRed	clrMediumOrchid	clrGreenYellow	clrMediumAquamarine	clrDarkSeaGreen	clrTomato	clrRosyBrown	clrOrchid
clrMediumPurple	clrPaleVioletRed	clrCoral	clrCornflowerBlue	clrDarkGray	clrSandyBrown	clrMediumSlateBlue	clrTan
clrDarkSalmon	clrBurlyWood	clrHotPink	clrSalmon	clrViolet	clrLightCoral	clrSkyBlue	clrLightSalmon
clrPlum	clrKhaki	clrLightGreen	clrAquamarine	clrSilver	clrLightSkyBlue	clrLightSteelBlue	clrLightBlue
clrPaleGreen	clrThistle	clrPowderBlue	clrPaleGoldenrod	clrPaleTurquoise	clrLightGray	clrWheat	clrNavajoWhite
clrMoccasin	clrLightPink	clrGainsboro	clrPeachPuff	clrPink	clrBisque	clrLightGoldenrod	clrBlanchedAlmond
clrLemonChiffon	clrBeige	clrAntiqueWhite	clrPapayaWhip	clrCornsilk	clrLightYellow	clrLightCyan	clrLinen
clrLavender	clrMistyRose	clrOldLace	clrWhiteSmoke	clrSeashell	clrIvory	clrHoneydew	clrAliceBlue
clrLavenderBlush	clrMintCream	clrSnow	clrWhite				

La couleur peut être définie sur un objet en utilisant la fonction ObjectSetInteger(). Pour définir la couleur des indicateurs personnalisés, utiliser la fonction PlotIndexSetInteger(). Pour obtenir des valeurs de couleur, il existe des fonctions similaires ObjectGetInteger() et PlotIndexGetInteger().

© 2000-2024, MetaQuotes Ltd.

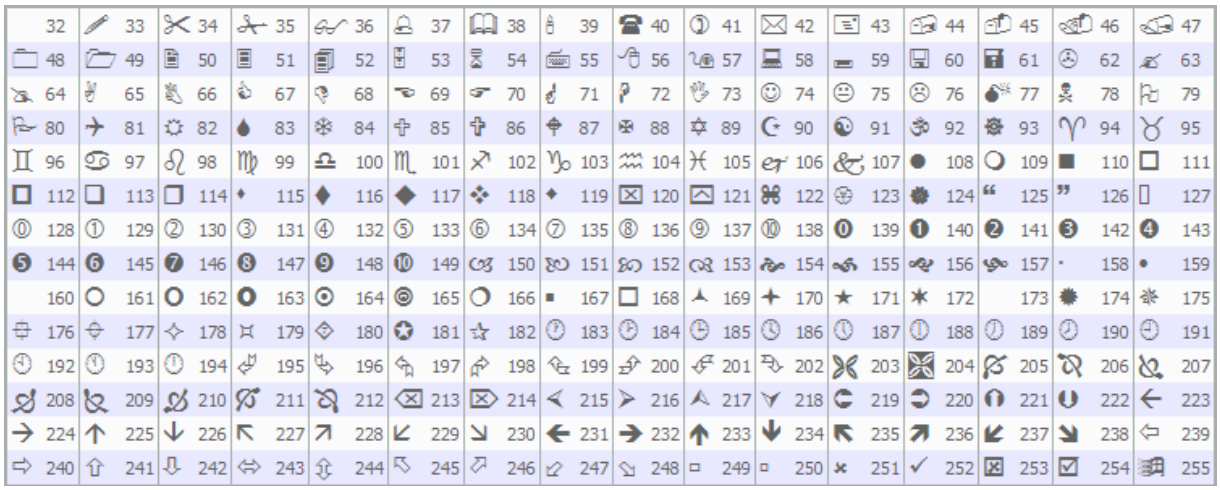
Exemple :

```
//---- paramètres de l'indicateur
#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_type3   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_color2  clrRed
#property indicator_color3  clrLime
```

© 2000-2024, MetaQuotes Ltd.

Wingdings

Les caractères Wingdings utilisés avec l'objet OBJ_ARROW :



Un caractère peut être définit en utilisant la fonction ObjectSetInteger().

Exemple :

```
void OnStart()
  {
//---
   string up_arrow="up_arrow";
   datetime time=TimeCurrent();
   double lastClose[1];
   int close=CopyClose(Symbol(),Period(),0,1,lastClose);      // Récupère le prix Close
//--- Si le prix est obtenu
   if(close>0)
     {
      ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);           // Crée une flêche
      ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);     // Définit le code de la
      ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);         // Définit l'heure
      ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]);// Définit le prix
      ChartRedraw(0);                                         // Déssine une flêche
     }
   else
      Print("Impossible d'obtenir le dernier prix de fermeture);
  }
```

© 2000-2024, MetaQuotes Ltd.

Constantes d'Indicateurs

Il existe 37 indicateurs techniques prédéfinis pouvant être utilisés dans les programmes écrits dans le langage MQL5. Il y a également la possibilité de créer des indicateurs techniques avec la fonction iCustom(). Toutes les constantes requises sont divisées en 5 groupes :

- Constantes de prix – pour sélectionner le type de prix ou de volume sur lequel un indicateur est calculé ;
- Méthodes de lissage – méthodes de lissage intégrées utilisées dans les indicateurs ;
- Lignes d'indicateur – identifiants des buffers de l'indicateur lors de l'accès à leurs valeurs en utilisant CopyBuffer() ;
- Styles de dessin – pour indiquer l'un des 18 types de dessin et pour définir le style de dessin des lignes ;
- Propriétés des indicateurs personnalisés sont utilisées pour travailler avec les indicateurs personnalisés ;
- Types d'indicateurs sont utilisés pour spécifier le type de l'indicateur technique lors de la création d'unhandle avec IndicatorCreate() ;
- Identifiants des types de données sont utilisés pour spécifier le type de donnée passé dans un tableau de données de type MqlParam dans la fonction IndicatorCreate().

© 2000-2024, MetaQuotes Ltd.

Constantes des Prix

Les indicateurs techniques ont besoin des prix et/ou des volumes pour effectuer les différents calculs. Il existe 7 identifiants prédéfinis dans l'énumération ENUM_APPLIED_PRICE, utilisés pour spécifier le prix désiré pour les calculs.

ENUM_APPLIED_PRICE

Identifiant	Description
PRICE_CLOSE	Prix de clôture (close)
PRICE_OPEN	Prix d'ouverture (open)
PRICE_HIGH	Le prix maximum pour la période
PRICE_LOW	Le prix minimum pour la période
PRICE_MEDIAN	Prix médian, (high + low)/2
PRICE_TYPICAL	Prix typique, (high + low + close)/3
PRICE_WEIGHTED	Prix moyen, (high + low + close + close)/4

Si le volume est utilisé pour les calculs, il est nécessaire de spécifier l'une des deux valeurs de l'énumération ENUM_APPLIED_VOLUME.

ENUM_APPLIED_VOLUME

Identifiant	Description
VOLUME_TICK	Volume des ticks
VOLUME_REAL	Volume des trades

L'indicateur technique iStochastic() peut être calculé de deux façons en utilisant :

- soit seulement les prix de Clôture (Close) ;
- ou les prix High et Low.

Pour sélectionner une variante nécessaire pour le calcul, spécifiez l'une des deux valeurs de l'énumération ENUM_STO_PRICE.

ENUM_STO_PRICE

Identifiant	Description
STO_LOWHIGH	Le calcul est basé sur les prix Low/High
STO_CLOSECLOSE	Le calcul est basé sur les prix Close/Close

Si un indicateur technique utilise les prix, dont le type est défini par ENUM_APPLIED_PRICE, pour les calculs alors le handle de n'importe quel indicateur (intégré du terminal ou écrit par l'utilisateur) peut être utilisé comme série de prix d'entrée. DAns ce cas, les valeurs du buffer zéro de l'indicateur sera utilisé pour les calculs. Cela simplifie la construction des valeurs d'un indicateur en utilisation les valeurs d'un autre indicateur. Le handle d'un indicateur personnalisé est créé en appelant la fonction iCustom().

© 2000-2024, MetaQuotes Ltd.

Exemple :

```
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- paramètres d'entrée
input int      RSIperiod=14;         // Période pour le calcul du RSI
input int      Smooth=8;             // Période de lissage du RSI
input ENUM_MA_METHOD meth=MODE_SMMA; // Méthode de lissage
//---- dessine le RSI
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- dessine le RSI lissé
#property indicator_label2  "RSI_Smoothed"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrNavy
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- buffers de l'indicateur
double         RSIBuffer[];          // Ici seront stockées les valeurs du RSI
double         RSI_SmoothedBuffer[]; // Ici seront stockées les valeurs lissées du RSI
int            RSIhandle;            // Handle du indicateur RSI
//+------------------------------------------------------------------------------------
//| Fonction d'initialisation d'un indicateur personnalisé
//+------------------------------------------------------------------------------------
void OnInit()
  {
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,RSIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,RSI_SmoothedBuffer,INDICATOR_DATA);
   IndicatorSetString(INDICATOR_SHORTNAME,"iRSI");
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//---
   RSIhandle=iRSI(NULL,0,RSIperiod,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------------------------
//| Fonction d'itération de l'indicateur personnalisé
//+------------------------------------------------------------------------------------
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[]
                )

  {
```

© 2000-2024, MetaQuotes Ltd.

```
//---  Réinitialise la valeur de la dernière erreur
   ResetLastError();
//--- Récupère les données de l'indicateur RSI dans le tableau RSIBuffer []
   int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);
   if(copied<=0)
     {
      Print("Impossible de copier les valeurs de l'indicateur RSI. Erreur = ",
            GetLastError(),",  copiés =",copied);
      return(0);
     }
//--- Crée les valeurs moyennées de l'indicateur en utilisant les valeurs du RSI
   int RSI_MA_handle=iMA(NULL,0,Smooth,0,meth,RSIhandle);
   copied=CopyBuffer(RSI_MA_handle,0,0,rates_total,RSI_SmoothedBuffer);
   if(copied<=0)
     {
      Print("Impossible de copier l'indicateur lissé du RSI. Erreur = ",
            GetLastError(),",  copiés =",copied);
      return(0);
     }
//--- retourne la valeur de prev_calculated pour l'appel suivant
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

Méthodes de Lissage

De nombreux indicateurs techniques sont basés sur différentes méthodes de lissage des séries de prix. Certaines indicateurs techniques standards nécessite de spécifier le type de lissage comme paramètre d'entrée. Pour spécifier le type désiré de lissage, il faut utiliser les identifiants listés dans l'énumération ENUM_MA_METHOD.

ENUM_MA_METHOD

Identifiant	Description
MODE_SMA	Moyenne simple
MODE_EMA	Moyenne exponentielle
MODE_SMMA	Moyenne lissée
MODE_LWMA	Moyenne pondérée linéaire

Exemple :

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//---- handles pour les moyennes mobiles
int    ExtJawsHandle;
int    ExtTeethHandle;
int    ExtLipsHandle;
//--- récupère les handles des MM
ExtJawsHandle=iMA(NULL,0,JawsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtTeethHandle=iMA(NULL,0,TeethPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtLipsHandle=iMA(NULL,0,LipsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
```

© 2000-2024, MetaQuotes Ltd.

Lignes des Indicateurs

Certains indicateurs techniques ont plusieurs buffers dessinés sur le graphique. La numérotation des buffers d'indicateur commence à 0. Lors de la copie des valeurs de l'indicateur avec la fonction CopyBuffer() dans un tableau de données de type double, pour quelques indicateurs il est possible d'indiquer l'identifiant d'un buffer copié au lieu de son numéro.

Identifiants possibles des lignes de l'indicateur lors de la copie des valeurs de iMACD(), iRVI() et iStochastic().

Constante	Valeur	Description
MAIN_LINE	0	Ligne principale
SIGNAL_LINE	1	Ligne de signal

Identifiants possibles des lignes de l'indicateur lors de la copie des valeurs de ADX() et ADXW().

Constante	Valeur	Description
MAIN_LINE	0	Ligne principale
PLUSDI_LINE	1	Ligne +DI
MINUSDI_LINE	2	Ligne -DI

Identifiants possibles des lignes de l'indicateur lors de la copie des valeurs de iBands().

Constante	Valeur	Description
BASE_LINE	0	Ligne principale
UPPER_BAND	1	Limite supérieure
LOWER_BAND	2	Limite inférieure

Identifiants possibles des lignes de l'indicateur lors de la copie des valeurs de iEnvelopes() et iFractals().

Constante	Valeur	Description
UPPER_LINE	0	Ligne supérieure
LOWER_LINE	1	Ligne du bas

Identifiants possibles des lignes de l'indicateur lors de la copie des valeurs de iGator()

Constante	Valeur	Description
UPPER_HISTOGRAM	0	Histogramme supérieur
LOWER_HISTOGRAM	2	Histogramme inférieur

Identifiants possibles des lignes de l'indicateur lors de la copie des valeurs de iAlligator().

© 2000-2024, MetaQuotes Ltd.

Constante	Valeur	Description
GATORJAW_LINE	0	Ligne de la mâchoire
GATORTEETH_LINE	1	Ligne des dents
GATORLIPS_LINE	2	Ligne des lèvres

Identifiants possibles des lignes de l'indicateur lors de la copie des valeurs de l'iIchimoku().

Constante	Valeur	Description
TENKANSEN_LINE	0	Ligne Tenkan-sen
KIJUNSEN_LINE	1	Ligne Kijun-sen
SENKOUSPANA_LINE	2	Ligne Senkou Span A
SENKOUSPANB_LINE	3	Ligne Senkou Span B
CHIKOUSPAN_LINE	4	Ligne Chikou Span

© 2000-2024, MetaQuotes Ltd.

Styles de Dessin

Lors de la création d'un indicateur personnalisé, vous pouvez spécifier l'un des 18 types de dessin (tel qu'affiché dans la fenêtre principale du graphique ou dans une sous-fenêtre), dont les valeurs sont spécifiées dans l'énumération ENUM_DRAW_TYPE.

Dans un indicateur personnalisé, il est possible d'utiliser n'importe quel type de dessin/construction d'indicateur. Chaque type de construction nécessite de spécifier de 1 à 5 tableaux globaux pour stocker les données nécessaires pour le dessin. Ces tableaux de données doivent être limités aux buffers des indicateurs avec la fonction SetIndexBuffer(). Le type de données de ENUM_INDEXBUFFER_TYPE doit être spécifié pour chaque buffer.

Suivant le style de dessin, il peut être nécessaire de spécifier de 1 à 4 buffers de valeurs (marqués avec INDICATOR_DATA). Si un style autorise une alternance de couleurs (tous les styles comporte COLOR dans leur nom), vous aurez alors besoin d'un buffer de couleur supplémentaire (type indiqué INDICATOR_COLOR_INDEX). Les buffers de couleurs sont toujours limités aux buffers de valeurs correspondant au style.

ENUM_DRAW_TYPE

Identifiant	Description	Buffers de données	Buffers de couleurs
DRAW_NONE	Non dessiné	1	0
DRAW_LINE	Ligne	1	0
DRAW_SECTION	Section	1	0
DRAW_HISTOGRAM	Histogramme depuis la ligne zéro	1	0
DRAW_HISTOGRAM2	Histogramme de deux buffers	2	0
DRAW_ARROW	Dessin de flèches	1	0
DRAW_ZIGZAG	Le style Zigzag permet de sectionner verticalement la barre	2	0

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Buffers de données	Buffers de couleurs
DRAW_FILLING	Couleur de remplissage entre deux niveaux	2	0
DRAW_BARS	Affichage sous la forme d'une suite de barres	4	0
DRAW_CANDLES	Affichage sous forme d'une suite de chandeliers	4	0
DRAW_COLOR_LINE	Ligne multi-couleurs	1	1
DRAW_COLOR_SECTION	Section multi-couleurs	1	1
DRAW_COLOR_HISTOGRAM	Histogramme multi-couleurs depuis la ligne zéro	1	1
DRAW_COLOR_HISTOGRAM2	Histogramme multi-couleurs des deux buffers de l'indicateur	2	1

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Buffers de données	Buffers de couleurs
DRAW_COLOR_ARROW	Dessin de flèches multi-couleurs	1	1
DRAW_COLOR_ZIGZAG	ZigZag multi-couleurs	2	1
DRAW_COLOR_BARS	Barres multi-couleurs	4	1
DRAW_COLOR_CANDLES	Chandeliers multi-couleurs	4	1

Pour affiner l'affichage du type de dessin sélectionné, les identifiants de ENUM_PLOT_PROPERTY sont utilisés.

Pour les fonctions PlotIndexSetInteger() et PlotIndexGetInteger()

ENUM_PLOT_PROPERTY_INTEGER

Identifiant	Description	Type de la Propriété
PLOT_ARROW	Code de la flèche à dessiner pour le style DRAW_ARROW	uchar
PLOT_ARROW_SHIFT	Décalage vertical pour les flèches pour le style DRAW_ARROW	int
PLOT_DRAW_BEGIN	Nombre de barres initiales sans dessin ni valeurs dans la Fenêtre des Données	int
PLOT_DRAW_TYPE	Type de construction graphique	ENUM_DRAW_TYPE
PLOT_SHOW_DATA	Signe d'affichage des valeurs de	bool

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
	construction dans la Fenêtre des Données	
PLOT_SHIFT	Décalage du dessin de l'indicateur sur l'axe du temps, en barres	int
PLOT_LINE_STYLE	Style de dessin de la ligne	ENUM_LINE_STYLE
PLOT_LINE_WIDTH	Epaisseur de la ligne de dessin	int
PLOT_COLOR_INDEXES	Nombre de couleurs	int
PLOT_LINE_COLOR	Indice d'un buffer contenant la couleur de dessin	color modificateur = numéro d'indice des couleurs

Pour la fonction PlotIndexSetDouble()

ENUM_PLOT_PROPERTY_DOUBLE

Identifiant	Description	Type de la Propriété
PLOT_EMPTY_VALUE	Valeur vide pour laquelle rien ne sera dessiné	double

Pour la fonction PlotIndexSetString()

ENUM_PLOT_PROPERTY_STRING

Identifiant	Description	Type de la Propriété
PLOT_LABEL	Le nom des séries graphiques de l'indicateur à afficher dans la Fenêtre des Données. L'utilisation de styles graphiques complexes nécessite plusieurs buffers pour l'affichage, les noms des	string

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
	buffers peuvent être spécifiés en utilisant ";" comme spérateur. Le code d'exemple est affiché dans DRAW_CANDLES	

5 styles peuvent être utilisés pour le dessin des lignes dans les indicateurs personnalisés. Ils ne sont valides que pour une épaisseur de ligne de 0 ou 1.

ENUM_LINE_STYLE

Identifiant	Description
STYLE_SOLID	Ligne solide
STYLE_DASH	Ligne brisée
STYLE_DOT	Ligne pointillée
STYLE_DASHDOT	Ligne tiret-point
STYLE_DASHDOTDOT	Tiret - deux points

Pour définir le style de ligne et le type de dessin, il faut utiliser la fonction PlotIndexSetInteger(). Pour l'expansion des Fibonacci, l'épaisseur et le style de dessin des niveaux peuvent être indiqués avec la fonction ObjectSetInteger().

Exemple :

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- buffers de l'indicateur
double         MABuffer[];
//+------------------------------------------------------------------------------------
//| Fonction d'initialisation d'un indicateur personnalisé
//+------------------------------------------------------------------------------------
void OnInit()
  {
//--- Lie le tableau du buffer de l'indicateur avec l'indice 0
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- Définit le style de dessin
   PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- Définit le style de ligne
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
//--- Définit la couleur de ligne
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Définit l'épaisseur de ligne
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
//--- Définit l'étiquette de la ligne
   PlotIndexSetString(0,PLOT_LABEL,"Moving Average");
//---
  }
//+------------------------------------------------------------------------------------
//| Fonction d'itération de l'indicateur personnalisé
//+------------------------------------------------------------------------------------
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   for(int i=prev_calculated;i<rates_total;i++)
     {
      MABuffer[i]=close[i];
     }
//--- retourne la valeur de prev_calculated pour l'appel suivant
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

Propriétés des Indicateurs Personnalisés

Le nombre de buffers pouvant être utilisés dans un indicateur personnalisé est illimité. Mais pour chaque tableau désigné comme buffer pour l'indicateur avec la fonction SetIndexBuffer(), il est nécessaire de spécifier le type des données qu'il contiendra. Cela peut être l'une des valeurs de l'énumération ENUM_INDEXBUFFER_TYPE.

ENUM_INDEXBUFFER_TYPE

Identifiant	Description
INDICATOR_DATA	Données à dessiner
INDICATOR_COLOR_INDEX	Couleur
INDICATOR_CALCULATIONS	Buffers supplémentaires pour les calculs intermédiaires

Un indicateur personnalisé doit fournir de nombreux paramètres pour un bon affichage. Ces paramètres sont définis avec l'assignation des propriétés correspondantes de l'indicateur avec les fonctions IndicatorSetDouble(), IndicatorSetInteger() et IndicatorSetString(). Les identifiants des propriétés de l'indicateur sont listés dans l'énumération ENUM_CUSTOMIND_PROPERTY.

ENUM_CUSTOMIND_PROPERTY_INTEGER

Identifiant	Description	Type de la Propriété
INDICATOR_DIGITS	Précision de dessin des valeurs de l'indicateur	int
INDICATOR_HEIGHT	Hauteur fixée de la fenêtr	int

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
	e de l'indicateur (commande du préprocesseur #property indicator_height)	
INDICATOR_LEVELS	Nombre de niveaux dans la fenêtre de l'indicateur	int
INDICATOR_LEVELCOLOR	Couleur de la	color modifier = numéro du niveau

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
	ligne du niveau	
INDICATOR_LEVELSTYLE	Style de la ligne du niveau	ENUM_LINE_STYLE modificateur = numéro du niveau
INDICATOR_LEVELWIDTH	Epaisseur de la ligne du niveau	int modificateur = numéro du niveau
INDICATOR_FIXED_MINIMUM	Minimum fixe pour la fenêtre de l'indicateur. La propriété ne peu	bool

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
	t être écrite que par la fonction IndicatorSetInteger()	
INDICATOR_FIXED_MAXIMUM	Maximum fixe pour la fenêtre de l'indicateur. La propriété ne peut être écrite que par la	bool

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
	fonction IndicatorSetInteger()	

ENUM_CUSTOMIND_PROPERTY_DOUBLE

Identifiant	Description	Type de la Propriété
INDICATOR_MINIMUM	Minimum de la fenêtre de l'indicateur	double
INDICATOR_MAXIMUM	Maximum de la fenêtre de l'indicateur	double
INDICATOR_LEVELVALUE	Valeur du	double modificateur = numéro du niveau

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type de la Propriété
	niveau	

ENUM_CUSTOMIND_PROPERTY_STRING

Identifiant	Description	Type de la Propriété
INDICATOR_SHORTNAME	Nom court de l'indicateur	string
INDICATOR_LEVELTEXT	Description du niveau	string modificateur = numéro du niveau

Exemples :

```
//--- paramètres de l'indicateur
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_color2  clrRed
//--- paramètres d'entrée
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- buffers de l'indicateur
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
double LowesBuffer[];
//+------------------------------------------------------------------------------
//| Fonction d'initialisation d'un indicateur personnalisé
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------------------------
void OnInit()
  {
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
   SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- définit la précision
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- définit les niveaux
   IndicatorSetInteger(INDICATOR_LEVELS,2);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- définit le maximum et le minimum pour la sous-fenêtre
   IndicatorSetDouble(INDICATOR_MINIMUM,0);
   IndicatorSetDouble(INDICATOR_MAXIMUM,100);
//--- définit la première barre à partir de laquelle l'indicateur sera dessiné
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
   PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- définit le style STYLE_DOT pour la seconde ligne
   PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- nom de la sous-fenêtre de l'indicateur et des données
   IndicatorSetString(INDICATOR_SHORTNAME,"Stoch("+KPeriod+","+DPeriod+","+Slowing+")"
   PlotIndexSetString(0,PLOT_LABEL,"Main");
   PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- définit la ligne de dessin à vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- initialisation terminée
  }
```

© 2000-2024, MetaQuotes Ltd.

Types des Indicateurs Techniques

Il y a deux façons de créer le handle d'un indicateur pour pouvroi accéder à ses valeurs. La première façon est de spécifier directement un nom de fonction de la liste des indicateurs techniques. La seconde méthode utilisant la fonction IndicatorCreate() permet de créer uniformément le handle de n'importe quel indicateur en assignant un identifiant de l'énumération ENUM_INDICATOR. Les deux façons de créer un handle sont égales, vous pouvez utiliser celle qui est la plus pratique suivant votre cas d'utilisation lors de l'écriture d'un programme MQL5.

Lors de la création d'un indicateur du type IND_CUSTOM, le champ *type* du premier élément d'un tableau de paramètres d'entrée MqlParam doit avoir la valeur TYPE_STRING de l'énumération ENUM_DATATYPE, tandis que le champ *string_value* du premier élément doit contenir le nom de l'indicateur personnalisé.

ENUM_INDICATOR

Identificateur	Indicateur
IND_AC	Oscillateur d'Accélération
IND_AD	Accumulation/Distribution
IND_ADX	Average Directional Index
IND_ADXW	ADX par Welles Wilder
IND_ALLIGATOR	Alligator
IND_AMA	Adaptive Moving Average
IND_AO	Awesome Oscillator
IND_ATR	Average True Range
IND_BANDS	Bandes de Bollinger®
IND_BEARS	Bears Power
IND_BULLS	Bulls Power
IND_BWMFI	Market Facilitation Index
IND_CCI	Commodity Channel Index
IND_CHAIKIN	Oscillateur de Chaikin
IND_CUSTOM	Indicateur personnalisér
IND_DEMA	Double Exponential Moving Average
IND_DEMARKER	DeMarker
IND_ENVELOPES	Enveloppes
IND_FORCE	Force Index
IND_FRACTALS	Fractals
IND_FRAMA	Fractal Adaptive Moving Average

© 2000-2024, MetaQuotes Ltd.

Identificateur	Indicateur
IND_GATOR	Gator Oscillator
IND_ICHIMOKU	Ichimoku Kinko Hyo
IND_MA	Moyenne Mobile (moving average)
IND_MACD	MACD
IND_MFI	Money Flow Index
IND_MOMENTUM	Momentum
IND_OBV	On Balance Volume
IND_OSMA	OsMA
IND_RSI	Relative Strength Index
IND_RVI	Relative Vigor Index
IND_SAR	Parabolic SAR
IND_STDDEV	Ecart-type
IND_STOCHASTIC	Oscillateur Stochastique
IND_TEMA	Triple Exponential Moving Average
IND_TRIX	Triple Exponential Moving Averages Oscillator
IND_VIDYA	Variable Index Dynamic Average
IND_VOLUMES	Volumes
IND_WPR	Williams' Percent Range

© 2000-2024, MetaQuotes Ltd.

Identifiants des Type de Données

Lors de la création du handle d'un indicateur avec la fonction IndicatorCreate(), un tableau de types MqlParam doit être spécifié comme dernier paramètre. De la même façon, la structure MqlParam, décrivant l'indicateur, contient un champ spécial *type*. Ce champ contient des informations sur le type de données (type real, integer ou string) qui est passée par un élément en particulier du tableau. La valeur de ce champ de la structure MqlParam peut être l'une des valeurs de ENUM_DATATYPE.

ENUM_DATATYPE

Identificateur	Type de donnée
TYPE_BOOL	bool
TYPE_CHAR	char
TYPE_UCHAR	uchar
TYPE_SHORT	short
TYPE_USHORT	ushort
TYPE_COLOR	color
TYPE_INT	int
TYPE_UINT	uint
TYPE_DATETIME	datetime
TYPE_LONG	long
TYPE_ULONG	ulong
TYPE_FLOAT	float
TYPE_DOUBLE	double
TYPE_STRING	string

Chaque élément du tableau décrit le paramètre d'entrée correspondant d'un indicateur technique créé, le type et l'ordre des éléments du tableau doivent donc être maintenus strictement en accord avec la description.

© 2000-2024, MetaQuotes Ltd.

Etat de l'Environnement

Les constantes décrivant l'état d'exécution actuel d'un programme mql5 sont divisées en groupes :

- Propriétés du terminal client – informations sur le terminal client ;
- Propriétés du programme MQL5 exécuté – propriétés du programme mql5 pour aider à contrôler son exécution ;
- Propriétés du symbole – informations sur un symbole ;
- Propriétés du compte – informations sur le compte de trading actuel ;
- Statistiques de Test – résultats du test d'un Expert Advisor.

© 2000-2024, MetaQuotes Ltd.

Propriétés du Terminal Client

Les informations du terminal client peuvent être obtenues avec 2 fonctions : TerminalInfoInteger() et TerminalInfoString(). Pour les paramètres, ces fonctions acceptent des valeurs de ENUM_TERMINAL_INFO_INTEGER et ENUM_TERMINAL_INFO_STRING respectivement.

ENUM_TERMINAL_INFO_INTEGER

Identifiant	Description	Type
TERMINAL_BUILD	Le numéro de build du terminal client	int
TERMINAL_COMMUNITY_ACCOUNT	Le flag indique la présence des données d'autorisation de la MQL5.community dans le terminal	bool
TERMINAL_COMMUNITY_CONNECTION	Connexion à la MQL5.community	bool
TERMINAL_CONNECTED	Connexion au serveur de trades	bool
TERMINAL_DLLS_ALLOWED	Permission d'utiliser des DLL	bool
TERMINAL_TRADE_ALLOWED	Permission d'effectuer des opérations de trading	bool
TERMINAL_EMAIL_ENABLED	Permission d'envoyer des e-mails en utilisant le serveur SMTP et le login spécifiés dans les paramètres du terminal	bool
TERMINAL_FTP_ENABLED	Permission d'envoyer des rapports en utilisant le serveur FTP et le login spécifiés dans les paramètres du terminal	bool
TERMINAL_NOTIFICATIONS_ENABLED	Permission d'envoyer des notifications vers un smartphone	bool
TERMINAL_MAXBARS	Le nombre maximum de barres sur le graphique	int
TERMINAL_MQID	Le flag indique la présence de l'identifiant MetaQuotes pour les notifications Push	bool
TERMINAL_CODEPAGE	Numéro de la page de code de la langue installée sur le terminal client	int
TERMINAL_CPU_CORES	Le nombre de coeurs CPU dans le système	int
TERMINAL_DISK_SPACE	Espace disque libre pour le répertoire MQL5\Files du terminal (agent), en Mo	int

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
TERMINAL_MEMORY_PHYSICAL	Mémoire physique du système, en Mo	int
TERMINAL_MEMORY_TOTAL	Mémoire disponible pour le processus du terminal (agent), en Mo	int
TERMINAL_MEMORY_AVAILABLE	Mémoire libre du processus du terminal (agent), en Mo	int
TERMINAL_MEMORY_USED	Mémoire utilisée par le terminal (agent), en Mo	int
TERMINAL_X64	Indication du terminal en 64 bits	bool
TERMINAL_OPENCL_SUPPORT	La version d'OpenCL supportée dans le format 0x00010002 = 1.2. "0" signifie qu'OpenCL n'est pas supporté	int
TERMINAL_SCREEN_DPI	La résolution d'affichage de l'écran est mesurée en nombre de Points Par Pource (PPP ou Dots Per Inch (DPI) en anglais). Avec cette valeur, vous pouvez déterminer la taille des objets graphiques afin qu'ils apparaissent de la même façon sur des écrans ayant des résolutions différentes.	int
TERMINAL_SCREEN_LEFT	La coordonnée gauche de l'écran virtuel. Un écran virtuel est un rectangle couvrant tous les écrans. Si le système a 2 écrans ordonnés de droite à gauche, alors la coordonnée gauche de l'écran virtuel peut être sur la bordure des 2 écrans.	int
TERMINAL_SCREEN_TOP	La coordonnée supérieure de l'écran virtuel	int
TERMINAL_SCREEN_WIDTH	Largeur du terminal	int
TERMINAL_SCREEN_HEIGHT	Hauteur du terminal	int
TERMINAL_LEFT	La coordonnée gauche du terminal relativement à l'écran virtuel	int
TERMINAL_TOP	La coordonnée supérieure du terminal relativement à l'écran virtuel	int

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
TERMINAL_RIGHT	La coordonnée droite du terminal relativement à l'écran virtuel	int
TERMINAL_BOTTOM	La coordonnée inférieure du terminal relativement à l'écran virtuel	int
TERMINAL_PING_LAST	La dernière valeur connue du ping à un serveur de trades en microsecondes. 1 seconde comporte 1 million de microsecondes	int
TERMINAL_VPS	Indication que le terminal est lancé sur un serveur d'Hébergement Virtuel MetaTrader (MetaTrader VPS)	bool
Identificateur de la touche	**Description**	
TERMINAL_KEYSTATE_LEFT	Etat de la touche "Flèche gauche"	int
TERMINAL_KEYSTATE_UP	Etat de la touche "Flèche vers le haut"	int
TERMINAL_KEYSTATE_RIGHT	Etat de la touche "Flèche droite"	int
TERMINAL_KEYSTATE_DOWN	Etat de la touche "Flèche vers le bas"	int
TERMINAL_KEYSTATE_SHIFT	Etat de la touche "Shift"	int
TERMINAL_KEYSTATE_CONTROL	Etat de la touche "Ctrl"	int
TERMINAL_KEYSTATE_MENU	Etat de la touche "Windows"	int
TERMINAL_KEYSTATE_CAPSLOCK	Etat de la touche "Verrouillage majuscules"	int
TERMINAL_KEYSTATE_NUMLOCK	Etat de la touche "Verrouillage numérique"	int
TERMINAL_KEYSTATE_SCRLOCK	Etat de la touche "Arrêt du défilement"	int
TERMINAL_KEYSTATE_ENTER	Etat de la touche "Entrée"	int
TERMINAL_KEYSTATE_INSERT	Etat de la touche "Ins"	int
TERMINAL_KEYSTATE_DELETE	Etat de la touche "Suppr"	int
TERMINAL_KEYSTATE_HOME	Etat de la touche "Début"	int
TERMINAL_KEYSTATE_END	Etat de la touche "Fin"	int
TERMINAL_KEYSTATE_TAB	Etat de la touche "Tab"	int

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
TERMINAL_KEYSTATE_PAGEUP	Etat de la touche "Page précédente"	int
TERMINAL_KEYSTATE_PAGEDOWN	Etat de la touche "Page suivante"	int
TERMINAL_KEYSTATE_ESCAPE	Etat de la touche "Echap"	int

L'appel à TerminalInfoInteger(TERMINAL_KEYSTATE_XXX) retourne le même code d'état d'une touche que la fonction GetKeyState() dans MSDN.

Exemple de calcul du facteur de mise à l'échelle :

```
//--- Création d'un bouton de 1.5 pouces de largeur sur un écran
int screen_dpi = TerminalInfoInteger(TERMINAL_SCREEN_DPI); // Récupère la résolution e
int base_width = 144;                                      // La largeur de base en po
int width      = (button_width * screen_dpi) / 96;         // Calcule la largeur du bo
...

//--- Calcul du facteur d'échelle en pourcentage
int scale_factor=(TerminalInfoInteger(TERMINAL_SCREEN_DPI) * 100) / 96;
//--- Utilisation du facteur d'échelle
width=(base_width * scale_factor) / 100;
```

Dans l'exemple ci-dessus, la ressource graphique apparaît de façon identique sur des écrans ayant des résolutions différentes. La taille des éléments de contrôle (boutons, boîtes de dialogue, etc.) correspond aux paramètres de personnalisation.

ENUM_TERMINAL_INFO_DOUBLE

Identifiant	Description	Type
TERMINAL_COMMUNITY_BALANCE	Solde sur la MQL5.community	double
TERMINAL_RETRANSMISSION	Pourcentage de paquets réseau retransmis dans le protocole TCP/IP pour toutes les applications et services en cours d'exécution sur un ordinateur donné. La perte de paquets se produit même sur des réseaux ultra-rapides et correctement configurés. Dans ce cas, il n'y a pas de confirmation de la livraison d'un paquet entre le destinataire et l'expéditeur, les paquets perdus sont donc renvoyés.	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	Ce n'est pas une indication de la qualité de la connexion entre un terminal particulier et un serveur de trades, puisque le pourcentage est calculé pour l'activité réseau en entier, incluant les activités système et en arrière plan. La valeur de TERMINAL_RETRANSMISSION est demandé au système d'exploitation toutes les minutes. Le terminal lui-même ne calcule pas cette valeur.	

Les opérations sur les fichiers ne peuvent être effectuées que dans 2 répertoires ; les chemins correspondants peuvent être obtenus en demandant les propriétés TERMINAL_DATA_PATH et TERMINAL_COMMONDATA_PATH.

ENUM_TERMINAL_INFO_STRING

Identifiant	Description	Type
TERMINAL_LANGUAGE	Langue du terminal	string
TERMINAL_COMPANY	Nom de la société	string
TERMINAL_NAME	Nom du terminal	string
TERMINAL_PATH	Répertoire à partir duquel le terminal est exécuté	string
TERMINAL_DATA_PATH	Répertoire dans lequel les données du terminal sont stockées	string
TERMINAL_COMMONDATA_PATH	Chemin commun de tous les terminaux installés sur un ordinateur	string
TERMINAL_CPU_NAME	Nom du CPU	string
TERMINAL_CPU_ARCHITECTURE	Architecture CPU	string
TERMINAL_OS_VERSION	Nom du système d'exploitation de l'utilisateur	string

Pour une meilleure compréhension des chemins, stockés dans les propriétés des paramètres TERMINAL_PATH, TERMINAL_DATA_PATH et TERMINAL_COMMONDATA_PATH, il est recommandé d'exécuter le script, qui retournera ces valeurs pour la copie actuelle du terminal client installée sur votre ordinateur

© 2000-2024, MetaQuotes Ltd.

Exemple : Script returnant les chemins du terminal client

```
//+------------------------------------------------------------------------------------
//|                                          Check_TerminalPaths.mq5
//|                        Copyright 2009, MetaQuotes Software Corp.
//|                                              https://www.mql5.com
//+------------------------------------------------------------------------------------
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//---
   Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
   Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
   Print("TERMINAL_COMMONDATA_PATH = ",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
  }
```

Le résultat de l'exécution du script apparaît dans le Journal des Experts sous la forme suivante :

Toolbox

Time	Source	Message
2018.06.29 09:28:27.253	Paths (EURUSD,H1)	TERMINAL_PATH = C:\Program Files\MetaTrader 5
2018.06.29 09:28:27.254	Paths (EURUSD,H1)	TERMINAL_DATA_PATH = C:\Users\smith\AppData\Roaming\MetaQuotes\...
2018.06.29 09:28:27.254	Paths (EURUSD,H1)	TERMINAL_COMMONDATA_PATH = C:\Users\smith\AppData\Roaming\MetaQ...

Trade | Exposure | History | News | Mailbox 7 | Calendar | Company | Market | Alerts | Signals | Code Base | Experts | J

© 2000-2024, MetaQuotes Ltd.

Propriété d'un Programme MQL5 en Cours d'Exécution

Pour obtenir des informations sur le programme mql5 en cours d'exécution, les constantes de ENUM_MQL_INFO_INTEGER et de ENUM_MQL_INFO_STRING sont utilisées.

Pour la fonction MQLInfoInteger

ENUM_MQL_INFO_INTEGER

Identificateur	Description	Type
MQL_HANDLES_USED	Le nombre actuel de descripteurs d'objets actifs. Ceux-ci incluent à la fois des objets dynamiques (créés via new) et non dynamiques, des variables globales/locales ou des membres de classe. Plus un programme utilise de descripteurs, plus il consomme de ressources.	int
MQL_MEMORY_LIMIT	Montant maximum possible de mémoire dynamique pour le programme MQL5 en Mo	int
MQL_MEMORY_USED	La taille de la mémoire utilisée par le programme MQL5 en Mo	int
MQL_PROGRAM_TYPE	Le type du programme mql5	ENUM_PROGRAM_TYPE
MQL_DLLS_ALLOWED	La permission d'utiliser les DLL **pour le programme en cours**	bool

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	d'exécution donné	
MQL_TRADE_ALLOWED	La permission d'effectuer des opérations de trading **pour le programme en cours d'exécution donné**	bool
MQL_SIGNALS_ALLOWED	La permission de modifier les Signaux **pour le programme en cours d'exécution donné**	bool
MQL_DEBUG	Le flag indiquant le mode de debug	bool
MQL_PROFILER	Le flag indiquant que le programme est en mode profilage du code	bool
MQL_TESTER	Le flag indiquant le processus de test	bool
MQL_FORWARD	Le flag indiquant que le programme est en cours d'exécution dans le test avancé	bool
MQL_OPTIMIZATION	Le flag indiquant le processus d'optimisation	bool
MQL_VISUAL_MODE	Le flag indiquant le processus de test visuel	bool
MQL_FRAME_MODE	Le flag indiquant que l'Expert Advisor est en train de	bool

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	rassembler les résultats d'opitmisation	
MQL_LICENSE_TYPE	Le type de licence du module EX5. La licence se réfère au module EX5 à partir duquel une demande est effectuée en utilisant la fonction MQLInfoInteger (MQL_LICENSE_TYPE).	ENUM_LICENSE_TYPE

Pour la fonction MQLInfoString

ENUM_MQL_INFO_STRING

Identificateur	Description	Type
MQL_PROGRAM_NAME	Nom du programme mql5 exécuté	string
MQL5_PROGRAM_PATH	Chemin **du programme en cours d'exécution donné**	string

Pour des informations sur le type du programme en cours d'exécution; les valeurs de ENUM_PROGRAM_TYPE sont utilisées.

ENUM_PROGRAM_TYPE

Identificateur	Description
PROGRAM_SCRIPT	Script
PROGRAM_EXPERT	Expert Advisor
PROGRAM_INDICATOR	Indicateur
PROGRAM_SERVICE	Service

© 2000-2024, MetaQuotes Ltd.

ENUM_LICENSE_TYPE

Identificateur	Description
LICENSE_FREE	Une version illimitée gratuite
LICENSE_DEMO	La version d'essai d'un produit payant du Market. Elle ne fonctionne que dans le testeur de stratégie
LICENSE_FULL	Une version avec licence achetée permet au moins 5 activations. Le nombre d'activations est spécifié par le vendeur. Le vendeur peut augmenter le nombre d'activations autorisées
LICENSE_TIME	Une version avec une licence limitée

Exemple :

```
ENUM_PROGRAM_TYPE mql_program=(ENUM_PROGRAM_TYPE)MQLInfoInteger(MQL_PROGRAM_TYPE);
switch(mql_program)
  {
   case PROGRAM_SCRIPT:
     {
      Print(__FILE__+" is script");
      break;
     }
   case PROGRAM_EXPERT:
     {
      Print(__FILE__+" is Expert Advisor");
      break;
     }
   case PROGRAM_INDICATOR:
     {
      Print(__FILE__+" is custom indicator");
      break;
     }
   default:Print("MQL5 type value is ",mql_program);
//---
Print("MQLInfoInteger(MQL_MEMORY_LIMIT)=", MQLInfoInteger(MQL_MEMORY_LIMIT), " MB")
Print("MQLInfoInteger(MQL_MEMORY_USED)=", MQLInfoInteger(MQL_MEMORY_USED), " MB");
Print("MQLInfoInteger(MQL_HANDLES_USED)=", MQLInfoInteger(MQL_HANDLES_USED), " hanc
```

© 2000-2024, MetaQuotes Ltd.

Propriétés du Symbole

Pour obtenir les informations actuelles du marché, il existe plusieurs fonctions : SymbolInfoInteger(), SymbolInfoDouble() et SymbolInfoString(). Le premier paramètre est le nom du symbole, les valeurs du deuxième paramètre de la fonction peuvent être l'un des identifiants de The first parameter is the symbol name, the values of the second function parameter can be one of the identifiers of ENUM_SYMBOL_INFO_INTEGER, ENUM_SYMBOL_INFO_DOUBLE et ENUM_SYMBOL_INFO_STRING.

Pour la fonction SymbolInfoInteger()

ENUM_SYMBOL_INFO_INTEGER

Identifiant	Description	Type
SYMBOL_SECTOR	Le secteur de l'économie auquel appartient l'actif	ENUM_SYMBOL_SECTOR
SYMBOL_INDUSTRY	L'industrie ou la branche économique à laquelle appartient	ENUM_SYMBOL_INDUSTRY

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	le symbole	
SYMBOL_CUSTOM	Le symbole est un symbole personnalisé - il a été créé de façon synthétique sur la base d'autres symboles depuis le Market Watch	bool

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	et/ou des sources de données externes	
SYMBOL_BACKGROUND_COLOR	La couleur d'arrière plan utilisée pour le symbole dans le Market Watch	color
SYMBOL_CHART_MODE	Le type de prix utilisé pour	ENUM_SYMBOL_CHART_MODE

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	générer les barres des symboles, par exemple Bid ou Last	
SYMBOL_EXIST	Un symbole avec ce nom existe déjà	bool
SYMBOL_SELECT	Le symbole est sélectionné dans le Market	bool

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	Watch	
SYMBOL_VISIBLE	Le symbole est visible dans le Market Watch. Certains symboles (principalement ceux avec des taux croisés nécessaire pour le calcul	bool

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	des besoins de marge ou de profit dans la devise du dépôt) sont sélectionnés automatiquement, mais généralement ne sont pas visibles dans le Mar	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	ket Watch. Pour être affichés, ces symboles doivent être sélectionnés explicitement.	
SYMBOL_SESSION_DEALS	Le nombre de transactions dans la session act	long

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	uelle	
SYMBOL_SESSION_BUY_ORDERS	Le nombre d'ordres d'Achat du moment	long
SYMBOL_SESSION_SELL_ORDERS	Le nombre d'ordres à la Vente du moment	long
SYMBOL_VOLUME	Le volume de la dernière transaction	long

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
SYMBOL_VOLUMEHIGH	Le volume maximal du jour	long
SYMBOL_VOLUMELOW	Le volume minimal du jour	long
SYMBOL_TIME	L'heure de la dernière cotation	datetime
SYMBOL_TIME_MSC	L'heure de la dernière cotation en millise	long

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	con des dep uis 197 0.0 1.0 1	
SYMBOL_DIGITS	Le no mb re de déc ima les	int
SYMBOL_SPREAD_FLOAT	Spr ead flot tan t	bool
SYMBOL_SPREAD	Spr ead en poi nts	int
SYMBOL_TICKS_BOOKDEPTH	Le no mb re ma xim al de de ma nde s indi qué es dan s le Mar	int

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	ketWatch. Pour les symboles qui n'ont pas de file d'attente de requêtes, la valeur est égale à zéro.	
SYMBOL_TRADE_CALC_MODE	Mode de calcul des prix contractuels	ENUM_SYMBOL_CALC_MODE

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
SYMBOL_TRADE_MODE	Type d'exécution de l'ordre	ENUM_SYMBOL_TRADE_MODE
SYMBOL_START_TIME	Date de début du symbole (habituellement utilisé pour les futures)	datetime
SYMBOL_EXPIRATION_TIME	Date de fin du symbole (habituellement util	datetime

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	isé pour les futures))	
SYMBOL_TRADE_STOPS_LEVEL	Indentation minimale en points depuis le prix de clôture actuel pour placer les ordres Stop	int
SYMBOL_TRADE_FREEZE_LEVEL	Distance pour geler les opé	int

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	ration de trading en points	
SYMBOL_TRADE_EXEMODE	Mode d'exécution de la transaction	ENUM_SYMBOL_TRADE_EXECUTION
SYMBOL_SWAP_MODE	Modèle de calcul du Swap	ENUM_SYMBOL_SWAP_MODE
SYMBOL_SWAP_ROLLOVER3DAYS	Jour de la semaine pour la facturation du	ENUM_DAY_OF_WEEK

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	swap sur 3 jours	
SYMBOL_MARGIN_HEDGED_USE_LEG	Calcul de la marge de hedging en utilisant la plus grande jambe (Buy ou Sell)	bool
SYMBOL_EXPIRATION_MODE	Flags des modes d'expiration autorisés	int

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	de l'ordre	
SYMBOL_FILLING_MODE	Flags des modes de remplissage autorisés de l'ordre	int
SYMBOL_ORDER_MODE	Flags pour les types d'ordres autorisés	int
SYMBOL_ORDER_GTC_MODE	Expiration des ordres Stop Loss et Take	ENUM_SYMBOL_ORDER_GTC_MODE

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	Profit, si SYMBOL_EXPIRATION_MODE=SYMBOL_EXPIRATION_GTC (Good till canceled)	
SYMBOL_OPTION_MODE	Type d'option	ENUM_SYMBOL_OPTION_MODE
SYMBOL_OPTION_RIGHT	Droit de l'option (Call/Put)	ENUM_SYMBOL_OPTION_RIGHT

Pour la fonction SymbolInfoDouble()

© 2000-2024, MetaQuotes Ltd.

ENUM_SYMBOL_INFO_DOUBLE

Identifiant	Description	Type
SYMBOL_BID	Offre - meilleure offre de vente	double
SYMBOL_BIDHIGH	Offre maximale du jour	double
SYMBOL_BIDLOW	Offre minimale du jour	double
SYMBOL_ASK	Demande - meilleure offre d'achat	double
SYMBOL_ASKHIGH	Demande maximale du jour	double
SYMBOL_ASKLOW	Demande minimale du jour	double
SYMBOL_LAST	Prix de la dernière transaction	double
SYMBOL_LASTHIGH	Prix Last maximal du jour	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
SYMBOL_LASTLOW	Prix Last minimal du jour	double
SYMBOL_VOLUME_REAL	Le volume de la dernière transaction	double
SYMBOL_VOLUMEHIGH_REAL	Le volume maximal du jour	double
SYMBOL_VOLUMELOW_REAL	Le volume minimal du jour	double
SYMBOL_OPTION_STRIKE	Le prix strike pour une option. Le prix auquel le détenteur d'une option peut acheter (pour une option Call) ou vendre (pour une option Put) l'actif	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	sous-jacent, et le vendeur de l'option de vendre ou d'acheter le montant correspondant de l'actif sous-jacent.	
SYMBOL_POINT	Valeur du point du symbole	double
SYMBOL_TRADE_TICK_VALUE	Valeur du SYMBOL_TRADE_TICK_VALUE_PROFIT	double
SYMBOL_TRADE_TICK_VALUE_PROFIT	Prix calculé du prix pour une position gagnante	double
SYMBOL_TRADE_TICK_VALUE_LOSS	Prix calculé du prix pour	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	une position perdante	
SYMBOL_TRADE_TICK_SIZE	Changement minimal de prix	double
SYMBOL_TRADE_CONTRACT_SIZE	Taille du contrat	double
SYMBOL_TRADE_ACCRUED_INTEREST	Intérêts courus- intérêts courus accumulés, c'est à dire la partie des intérêts du coupon calculés en proportion du nombre de jours depuis l'émission du coupon de l'obligation ou depuis le dernier paieme	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	nt des intérêts du coupon	
SYMBOL_TRADE_FACE_VALUE	Valeur faciale - valeur initiale de l'obligation définie par l'émetteur	double
SYMBOL_TRADE_LIQUIDITY_RATE	Le Taux de Liquidité est la partie de l'actif pouvant être utilisée pour la marge.	double
SYMBOL_VOLUME_MIN	Volume minimal pour une transaction	double
SYMBOL_VOLUME_MAX	Volume maximal pour une transaction	double
SYMBOL_VOLUME_STEP	Pas de changement	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	minimal pour l'exécution d'une transaction	
SYMBOL_VOLUME_LIMIT	Volume agregé maximum autorisé d'une position ouverte et des ordres en attente dans une direction (buy ou sell) pour le symbole. Par exemple, avec une limite de 5 lots, vous pouvez avoir une position d'achat ouverte avec un volume de 5	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	lots et placer un ordre en attente Sell Limit avec un volume de 5 lots. Mais dans ce cas, vous ne pouvez pas placer un ordre en attente Buy Limit (puisque le volume total dans une direction serait supérieur à la limite) ou un ordre Sell Limit avec un volume supérie	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	ur à 5 lots.	
SYMBOL_SWAP_LONG	Valeur du swap long	double
SYMBOL_SWAP_SHORT	Valeur du swap court	double
SYMBOL_SWAP_SUNDAY	Ratio de calcul du swap (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) pour les positions de nuit reportées du DIMANCHE au jour suivant. Les valeurs suivantes sont prises en charge : • 0 - au	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	cunéchangen'estfacturé • 1-échangeunique • 3-échangeet	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	r i p l e	
SYMBOL_SWAP_MONDAY	Taux de calcul du swap (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) pour les positions de nuit reconduites du lundi au mardi	double
SYMBOL_SWAP_TUESDAY	Taux de calcul du swap (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) pour les positions de nuit	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	recondu ites du mardi au mercredi	
SYMBOL_SWAP_WEDNESDAY	Taux de calcul du swap (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) pour les positions de nuit recondu ites du mercredi au jeudi	double
SYMBOL_SWAP_THURSDAY	Taux de calcul du swap (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) pour les positio	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	ns de nuit recondu ites du jeudi au vendredi	
SYMBOL_SWAP_FRIDAY	Taux de calcul du swap (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) pour les positions de nuit reconduites du vendredi au samedi	double
SYMBOL_SWAP_SATURDAY	Taux de calcul du swap (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) pour	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	les positions de nuit reportées du samedi au dimanche	
SYMBOL_MARGIN_INITIAL	La marge initiale signifie la quantité dans la devise de marge requise pour ouvrir une position avec un volume d'un lot. Elle est utilisée pour vérifier les fonds d'un client lorsqu'il ou elle entre sur le marché. La fonctio	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	n SymbolInfoMarginRate() fournit des données sur le montant de la marge facturée en fonction du type et de la direction de l'ordre.	
SYMBOL_MARGIN_MAINTENANCE	La marge de maintien. Si elle est définie, elle détermine la quantité de la marge, dans la devise de marge du symbole, facturée pour un lot. Elle est utilisée pour vérifier	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	les fonds d'un client lorsque l'état de son compte change. Si la marge de maintien est égale à 0, la marge initiale est utilisée. La fonction SymbolInfoMarginRate() fournit des données sur le montant de la marge facturée en fonction du type et de la direction de l'ordre.	
SYMBOL_SESSION_VOLUME	Résumé du volume	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	de transactions en cours	
SYMBOL_SESSION_TURNOVER	Chiffre d'affaires de la session en cours	double
SYMBOL_SESSION_INTEREST	Intérêts de la session en cours	double
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Volume en cours des ordres d'Achat	double
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Volume en cours des ordres de Vente	double
SYMBOL_SESSION_OPEN	Prix d'ouverture de la session en cours	double
SYMBOL_SESSION_CLOSE	Prix de fermeture de la session en cours	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
SYMBOL_SESSION_AW	Prix moyen pondéré de la session en cours	double
SYMBOL_SESSION_PRICE_SETTLEMENT	Prix de règlement de la session en cours	double
SYMBOL_SESSION_PRICE_LIMIT_MIN	Prix minimal de la session en cours	double
SYMBOL_SESSION_PRICE_LIMIT_MAX	Prix maximal de la session en cours	double
SYMBOL_MARGIN_HEDGED	Taille de contrat ou valeur de marge pour un lot des positions hedgées (positions dans des directions opposé	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	es pour un symbole). Deux méthodes de calcul de la marge sont possibles pour les positions hedgées. La méthode de calcul est définie par le courtier. Calcul de base : • Si la marge initiale (SYMBOL_MARGIN_INITIAL) est spécifiée pour un symbole,	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	la marge hedgée est spécifiée en valeur absolue (en termes monétaires). • Si la marge initiale n'est pas spécifiée (égale à 0), SYMBOL_MARGIN_HEDGED est égal à la taille du contrat qui sera utilisé	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	pour calculer la marge avec la formule appropriée selon le type de l'instrument financier (SYMBOL_TRADE_CALC_MODE). Calcul pour la plus grande position : • La valeur SYMBOL_MARGIN_HEDGED n'est pas prise en	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	compte. • Le volume de toutes les positions courtes et longues d'un symbole est calculé. • Pour chaque direction, un prix d'ouverture moyen pondéré et un taux de conversion moyen pondéré dans la devise du	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	dépôt sont calculés. • La marge est ensuite calculée pour les parties longues et courtes, selon la formule appropriée choisie selon le type du symbole (SYMBOL_TRADE_CALC_MODE). • La plus grande des valeu	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	rs est utilisée comme marge.	
SYMBOL_PRICE_CHANGE	Change of the current price relative to the end of the previous trading day in %	double
SYMBOL_PRICE_VOLATILITY	Price volatility in %	double
SYMBOL_PRICE_THEORETICAL	Theoretical option price	double
SYMBOL_PRICE_DELTA	Option/warrant delta shows the value the option price changes by, when the underlying asset price	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	changes by 1	
SYMBOL_PRICE_THETA	Option/warrant theta shows the number of points the option price is to lose every day due to a temporary breakup, i.e. when the expiration date approaches	double
SYMBOL_PRICE_GAMMA	Option/warrant gamma shows the change rate of delta - how quickly or slowly the option premium	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	changes	
SYMBOL_PRICE_VEGA	Option/warrant vega shows the number of points the option price changes by when the volatility changes by 1%	double
SYMBOL_PRICE_RHO	Option/warrant rho reflects the sensitivity of the theoretical option price to the interest rate changing by 1%	double
SYMBOL_PRICE_OMEGA	Option/warrant omega. Option elasticity	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	shows a relative percentage change of the option price by the percentage change of the underlying asset price	
SYMBOL_PRICE_SENSITIVITY	Option/warrant sensitivity shows by how many points the price of the option's underlying asset should change so that the price of the option changes by one point	double

© 2000-2024, MetaQuotes Ltd.

Pour la fonction SymbolInfoString()

ENUM_SYMBOL_INFO_STRING

Identifiant	Description	Type
SYMBOL_BASIS	L'actif sous-jacent d'un dérivé	string
SYMBOL_CATEGORY	Le nom du secteur ou de la catégorie à laquelle le symbole de trading appartient	string
SYMBOL_COUNTRY	Le pays auquel appartient le symbole financier	string
SYMBOL_SECTOR_NAME	Le secteur de l'économie auquel appartient le symbole financier	string
SYMBOL_INDUSTRY_NAME	La branche industrielle ou l'industrie à laquelle appartient le symbole financier	string

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
SYMBOL_CURRENCY_BASE	Devise de base d'un symbole	string
SYMBOL_CURRENCY_PROFIT	Profit de devise	string
SYMBOL_CURRENCY_MARGIN	Marge de devise	string
SYMBOL_BANK	Fournisseur des cotations actuelles	string
SYMBOL_DESCRIPTION	Description du symbole	string
SYMBOL_EXCHANGE	Le nom de la bourse où le symbole financier est tradé	string
SYMBOL_FORMULA	La formule utilisée pour le calcul du prix du symbole personnalisé. Si le nom du symbole financier utilisé dans la formule commence avec un chiffre ou contient un caractère spécial (">" ", "." , "-",	string

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	"&", "#" etc), des guillemets doivent être utilisés autour du nom de ce symbole. • Symbole synthétique: "@ESU19"/EURCAD • Spre	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	adcalendaire: "Si-9.13" - "Si-6.13" • Euroindex: 34.388	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	0 5 7 2 6 * p o w (E U R U S D , 0 . 3 1 5 5) * p o w (E U R G B P , 0 . 3 0 5 6) * p o w (	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	EURJPY,0.1891)*pow(EURCHF,0.1113)*pow(EURSEK,0.0	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	785)	
SYMBOL_ISIN	Le nom du symbole dans le système ISIN (International Securities Identification Number). L'International Securities Identification Number est un numéro à 12 caractères alphanumériques qui identifie un titre de façon unique. La présence de cette propriété du symbole est définie au niveau du serveur	string

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
	de trades.	
SYMBOL_PAGE	L'adresse de la page web contenant les informations du symbole. Cette adresse sera affichée sous forme de lien lors de l'affichage des propriétés du symbole dans le terminal	string
SYMBOL_PATH	Chemin dans l'arbre des symboles	string

Le graphique des prix d'un symbole peut être basé sur les prix Bid ou Last. Le prix sélectionné pour les graphiques du symbole affecte également la génération et l'affichage des barres dans le terminal. Les valeurs possibles pour la propriété SYMBOL_CHART_MODE sont décrites dans l'énumération ENUM_SYMBOL_CHART_MODE

ENUM_SYMBOL_CHART_MODE

Identifiant	Description
SYMBOL_CHART_MODE_BID	Les barres sont basées sur les prix Bid
SYMBOL_CHART_MODE_LAST	Les barres sont basées sur les prix Last

© 2000-2024, MetaQuotes Ltd.

Pour chaque symbole, plusieurs modes d'expiration des ordres en attente peuvent être spécifiés. Un flag correspond à chaque mode. Les flags peuvent être combinés avec l'opérateur logique **OU** (|), par exemple, SYMBOL_EXPIRATION_GTC|SYMBOL_EXPIRATION_SPECIFIED. Pour vérifier si un certain mode est autorisé pour le symbole, le résultat du **ET** (&) logique doit être comparé avec le flag des modes.

Si le flag SYMBOL_EXPIRATION_SPECIFIED est spécifié pour un symbole, alors lors de l'envoi d'un ordre en attente, vous pouvez spécifier jusqu'à quand l'ordre en attente sera valide.

Identifiant	Valeur	Description
SYMBOL_EXPIRATION_GTC	1	L'ordre est valide pendant une durée illimitée, jusqu'à ce qu'il soit explicitement annulé
SYMBOL_EXPIRATION_DAY	2	L'ordre est valide jusqu'à la fin du jour
SYMBOL_EXPIRATION_SPECIFIED	4	L'heure d'expiration est spécifiée dans l'ordre
SYMBOL_EXPIRATION_SPECIFIED_DAY	8	La date d'expiration est spécifiée dans l'ordre

Exemple :

```
//+------------------------------------------------------------------------------------
//| Vérifie si le mode d'expiration spécifié est autorisé
//+------------------------------------------------------------------------------------
bool IsExpirationTypeAllowed(string symbol,int exp_type)
  {
//--- Récupère la valeur de la propriété décrivant les modes d'expiration autorisés
   int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- Retourne true si le mode exp_type est autorisé
   return((expiration&exp_type)==exp_type);
  }
```

Si la propriété SYMBOL_EXPIRATION_MODE est définie à SYMBOL_EXPIRATION_GTC (good till canceled), l'expiration des ordres en attente et des ordres Stop Loss/Take Profit doit être aussi définie avec l'énumération ENUM_SYMBOL_ORDER_GTC_MODE.

ENUM_SYMBOL_ORDER_GTC_MODE

Identifiant	Description
SYMBOL_ORDERS_GTC	Les niveaux des ordres en attente et des ordres Stop Loss/Take Profit sont valides pour une période illimitée jusqu'à leur annulation explicite
SYMBOL_ORDERS_DAILY	Les ordres ne sont valides que pendant une seule journée de trading. A la fin du jour, tous les

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description
	niveaux des Stop Loss et des Take Profit, ainsi que ceux des ordres en attente, sont supprimés.
SYMBOL_ORDERS_DAILY_EXCLUDING_STOPS	Lors du changement de la journée de trading, seuls les ordres en attente sont supprimés, tandis que les niveaux des Stop Loss et des Take Profit sont conservés.

Lors de l'envoi d'un ordre, nous pouvons spécifier la politique de remplissage du volume défini dans l'ordre. Les options d'exécution d'ordre possibles basées sur le volume pour chaque symbole sont spécifiées dans le tableau. Il est possible de définir plusieurs modes pour chaque instrument via une combinaison de drapeaux. La combinaison d'indicateurs est exprimée par l'opération logique **OU** (|), par exemple SYMBOL_FILLING_FOK|SYMBOL_FILLING_IOC. Pour vérifier si un mode spécifique est autorisé pour un instrument, comparez le résultat logique **ET** (&) avec l'indicateur de mode - exemple.

Politique de remplissage	ID	Valeur	Description
Fill or Kill	SYMBOL_FILLING_FOK	1	Un ordre ne peut être exécu

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			téquedansleevolumespécifié. Silemontantné

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			cessaire d'un instrument financier restactuel

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			lementindisponiblesurlemarché, l'ordrerenesse

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			rapas exécuté. Le volume souhaité peut être recom

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			posédeplusieursoffresdisponibles. Lorsde

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			l'envoi d'un ordre, le type de remplissage ORDER

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			_FILLING_FOK doit être spécifié pour cette pol

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			itique. La possibilité d'utiliser des ordres

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			FOK est déterminée sur les serveurs de trading.
Immediate or Cancel	SYMBOL_FILLING_IOC	2	Un

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			traders' engage à exécuter une transaction avec

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			clevolumemaximumdisponiblesurlemarchédan

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			s l a l i m i t e d e c e l u i i n d i q u é d a n s l' o r d r e . S i l a d e

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			mandenepeutpasêtreentièrementsatisfaite,

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			unordre avec le volume disponible sera exécuté

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			et le volume restant sera annulé. Lors de l'envo

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			id' unordre, le type de remplissage ORDER_FILL

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			ING_IOC doit être spécifié pour cette politique

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			e. La possibilité d'utiliser les ordres IOCes

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			tdéterminée au niveau du serveur de trading.
Passif	SYMBOL_FILLING_BOC	4	La

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			politiqueBOC (Book-or-Cancel) considère que

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			l'ordre ne peut qu'être placé dans le Depth of Mar

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			ketetnepeutpasêtreexécutéimmédiatement.S

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			il'ordre peut être exécuté immédiatement après

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			avoir été passé, il est alors annulé. En effet,

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			cette politique d'exécution ne peut être précisée

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			séequelorsquelepridelordrepassédoitêtre

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			inférieur à celui du marché actuel. Les ordres B

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			Ordres utilisés pour mettre en œuvre le trading

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			passif, desorte que l'ordre n'est pas exécuté im

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			médiatement lorsqu'il est placé et n'affecte p

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			aslaliquiditéactuelle. Seulslesordreslim

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			i t e t s t o p l i m i t s o n t p r i s e n c h a r g e , c ' e s t - à - d i

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			requel'indicateur SYMBOL_ORDER_MODE doitcon

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			tenir les valeurs SYMBOL_ORDER_LIMIT et/ou SY

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			MBOL_ORDER_STOP_LIMIT.
Retour	Pas d'identifiant		Encas de remplissage

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			partiel, un ordre au marché ou à cours limité avec

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			clevolumerestantn' estpasannulémaistraité

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			ultérieurement. Lors de l'envoi d'une ordre, le

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			type de remplissage ORDER_FILLING_RETURN doi

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			être spécifié pour cette politique. Les ordr

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			esdeerretourrnesontpasautorisésenmodded'exéc

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			utiondemarché(exécutiondemarché_SYMBOL_T

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	ID	Valeur	Description
			RADE_EXECUTION_MARKET).

Lors de l'envoi d'une demande de trade à l'aide de la fonction OrderSend(), la politique d'exécution du volume nécessaire peut être définie dans le champ *type_filling* dans la structure MqlTradeRequest. Les valeurs de l'énumération ENUM_ORDER_TYPE_FILLING sont disponibles. Si aucun type de remplissage n'est spécifié, ORDER_FILLING_RETURN est automatiquement défini dans la demande de trade. Le type de remplissage ORDER_FILLING_RETURN est activé dans n'importe quel mode d'exécution à l'exception de "Exécution au marché" (SYMBOL_TRADE_EXECUTION_MARKET).

Lors de l'envoi d'une demande de transaction pour exécution **à l'heure actuelle** (heure en vigueur), nous devons garder à l'esprit que les marchés financiers ne garantissent pas que l'intégralité du volume demandé est disponible pour un certain instrument financier au prix souhaité. Par conséquent, les opérations de trading en temps réel sont régulées à l'aide des modes d'exécution prix et volume. Les modes, ou politiques d'exécution, définissent les règles pour les cas où le prix a changé ou le volume demandé ne peut pas être complètement rempli **pour le moment**.

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	Une valeur de ENUM_SYMBOL_TRADE_EXECUTION
Mode d'exécution (Exécution de la demande)	Exécuter un ordre de marché au prix précédemment reçu du courtier. Les prix d'un certain ordre au marché sont demandés au courtier avant l'envoi de l'ordre. Dès réception des prix, l'exécution de l'ordre au prix indiqué peut être soit confirmée, soit refusée.	SYMBOL_TRADE_EXECUTION_REQUEST
Instant Execution (Exécution instantanée)	Exécuter immédi	SYMBOL_TRADE_EXECUTION_INSTANT

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	Une valeur de ENUM_SYMBOL_TRADE_EXECUTION
	atement un ordre de marché au prix spécifié. Lors de l'envoi d'une demande de trade à exécuter, la plateforme ajoute automatiquement les prix actuels à l'ordre. • Si le courtier accepte le	

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	Une valeur de ENUM_SYMBOL_TRADE_EXECUTION
	prix, l'ordre est exécuté. • Si le courtier n'accepte pas le	

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	Une valeur de ENUM_SYMBOL_TRADE_EXECUTION
	prix demandé, une "Recotation" est envoyée-le courtier	

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	Une valeur de ENUM_SYMBOL_TRADE_EXECUTION
	renvoie les prix auxquels ce type ordre peut être exécuté.	

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	Une valeur de ENUM_SYMBOL_TRADE_EXECUTION
Market Execution (Exécution du marché)	Un courtier prend une décision sur le prix d'exécution de l'ordre sans aucune discussion supplémentaire avec le trader. L'envoi de l'ordre dans un tel mode signifie le consentement préalable à son exécution à ce prix.	SYMBOL_TRADE_EXECUTION_MARKET
Exchange Execution (Exécution boursière)	Les opérations de trading sont exécutées aux prix des offres de	SYMBOL_TRADE_EXECUTION_EXCHANGE

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	Une valeur de ENUM_SYMBOL_TRADE_EXECUTION
	marché en cours.	

Avant d'envoyer un ordre avec l'heure d'exécution actuelle, pour le réglage correct de la valeur ORDER_TYPE_FILLING (type d'exécution du volume), vous pouvez utiliser la fonction SymbolInfoInteger() avec chaque instrument financier pour obtenir la valeur de la propriété SYMBOL_FILLING_MODE, qui montre les types d'exécution de volume autorisés pour le symbole en tant que combinaison de drapeaux. Le type de remplissage ORDER_FILLING_RETURN est activé à tout moment sauf pour le mode "Exécution au marché" (SYMBOL_TRADE_EXECUTION_MARKET).

L'utilisation des types de remplissage en fonction du mode d'exécution peut être illustrée dans le tableau suivant :

Type d'exécution\\Politique de remplissage	Fill or Kill (FOK ORDER_FILLING_FOK)	Immediate or Cancel (IOC ORDER_FILLING_IOC)	Return (Return ORDER_FILLING_RETURN)
Instant Execution (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (quel que soit le réglage du symbole)	+ (quel que soit le réglage du symbole)	+ (toujours)
Exécution de la Requête SYMBOL_TRADE_EXECUTION_REQUEST	+ (quel que soit le réglage du symbole)	+ (quel que soit le réglage du symbole)	+ (toujours)
Market Execution SYMBOL_TRADE_EXECUTION_MARKET	+ (défini dans les paramètres des symboles)	+ (défini dans les paramètres des symboles)	- (désactivé quels que soient les paramètres des symboles)
Exchange Execution SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (défini dans les paramètres des symboles)	+ (défini dans les paramètres des symboles)	+ (toujours)

En cas d'ordres en attente, le type de remplissage ORDER_FILLING_RETURN doit être utilisé quel que soit le type d'exécution (SYMBOL_TRADE_EXEMODE), car ces ordres ne sont pas destinés à être exécutés au moment de l'envoi. Lors de l'utilisation d'ordres en attente, un trader accepte à l'avance que, lorsque les conditions d'une transaction sur cet ordre sont remplies, le courtier utilisera le type de remplissage pris en charge par la bourse.

Exemple :

```
//+------------------------------------------------------------------+
//| vérifie si un mode de remplissage donné est autorisé             |
//+------------------------------------------------------------------+
bool IsFillingTypeAllowed(string symbol,int fill_type)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- récupère la valeur de la propriété décrivant le mode de remplissage
   int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- renvoie 'true' si le mode fill_type est autorisé
   return((filling&fill_type)==fill_type);
  }
```

Lors de l'envoi d'une demande de trade avec la fonction OrderSend(), un type d'ordre de l'énumération ENUM_ORDER_TYPE devrait être spécifié pour certaines opérations. Tous les types d'ordres ne sont pas autorisés pour un symbole donné. La propriété SYMBOL_ORDER_MODE décrit les flags des types d'ordres autorisés.

Identifiant	Valeur	Description
SYMBOL_ORDER_MARKET	1	Les ordres au marché sont autorisés (Buy et Sell)
SYMBOL_ORDER_LIMIT	2	Les ordres Limit sont autorisés (Buy Limit et Sell Limit)
SYMBOL_ORDER_STOP	4	Les ordres Stop sont autorisés (Buy Stop et Sell Stop)
SYMBOL_ORDER_STOP_LIMIT	8	Les ordres Stop-limit sont autorisés (Buy Stop Limit et Sell Stop Limit)
SYMBOL_ORDER_SL	16	Le Stop Loss est autorisé
SYMBOL_ORDER_TP	32	Le Take Profit est autorisé
SYMBOL_ORDER_CLOSEBY	64	Permission d'une opération Close By, c'est à dire la fermeture d'une position par une autre position ouverte sur le même instrument mais dans la direction opposée. La propriété est activée pour les comptes ayant le hedging (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)

Exemple :

```
//+------------------------------------------------------------------------------------
//| La fonction affiche les types d'ordres autorisés pour un symbole
//+------------------------------------------------------------------------------------
void Check_SYMBOL_ORDER_MODE(string symbol)
  {
//--- Récupère la valeur de la propriété décrivant les types d'ordres autorisés
   int symbol_order_mode=(int)SymbolInfoInteger(symbol,SYMBOL_ORDER_MODE);
//--- vérifie les ordres au marché (Exécution au Marché)
   if((SYMBOL_ORDER_MARKET&symbol_order_mode)==SYMBOL_ORDER_MARKET)
      Print(symbol+": Les ordres au Marché sont autorisés (Buy et Sell)");
```

© 2000-2024, MetaQuotes Ltd.

```
//--- vérifie les ordres Limit
   if((SYMBOL_ORDER_LIMIT&symbol_order_mode)==SYMBOL_ORDER_LIMIT)
      Print(symbol+": Les ordres Buy Limit et Sell Limit sont autorisés");
//--- vérifie les ordres Stop
   if((SYMBOL_ORDER_STOP&symbol_order_mode)==SYMBOL_ORDER_STOP)
      Print(symbol+": Les ordres Buy Stop et Sell Stop sont autorisés");
//--- vérifie les ordres Stop Limit
   if((SYMBOL_ORDER_STOP_LIMIT&symbol_order_mode)==SYMBOL_ORDER_STOP_LIMIT)
      Print(symbol+": Les ordres Buy Stop Limit et Sell Stop Limit sont autorisés");
//--- vérifie si le placement des ordres Stop Loss est autorisé
   if((SYMBOL_ORDER_SL&symbol_order_mode)==SYMBOL_ORDER_SL)
      Print(symbol+": Les ordres Stop Loss sont autorisés");
//--- vérifie si le placement des ordres Take Profit sont autorisés
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_TP)
      Print(symbol+": Les ordres Take Profit sont autorisés");
// --- vérifie si la fermeture d'une position par une position opposée est autorisée
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_CLOSEBY)
      Print(symbol+": Close by autorisé");
//---
  }
```

L'énumération ENUM_SYMBOL_CALC_MODE est utilisée pour obtenir des informations sur la façon dont les besoins de marge pour un symbole sont calculés.

ENUM_SYMBOL_CALC_MODE

Identifiant	Description	Formule
SYMBOL_CALC_MODE_FOREX	Mode Forex - calcul du profit et de la marge pour le Forex	Marge : Lots * Taille_Contrat/Levier * Margin_Rate Profit : (prix_clôture - prix_ouverture) * Taille_Contrat * Lots
SYMBOL_CALC_MODE_FOREX_NO_LEVERAGE	Mode Forex Aucu	Marge : Lots * Taille_Contrat * Margin_Rate Profit : (prix_clôture - prix_ouverture) * Taille_Contrat * Lots

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	n levier - calcul du profit et de la marge pour les symboles du Forex sans prendre en compte le levier	
SYMBOL_CALC_MODE_FUTURES	Mode Futures - calcul de la marge et du profit pour les futures	Marge : Lots * MargeInitiale * Margin_Rate Profit : (prix_clôture - prix_ouverture)*PrixTick / TailleTick * Lots
SYMBOL_CALC_MODE_CFD	Mode CFD - calcul de la	Marge : Lots * TailleContrat * PrixMarché * Margin_Rate Profit : (prix_clôture - prix_ouverture) * Taille_Contrat * Lots

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	marge et du profit pour les CFD	
SYMBOL_CALC_MODE_CFDINDEX	Mode indice CFD - calcul de la marge et du profitpour les indices CFD	Marge : (Lots * Taille_Contrat * PrixMarché) * PrixTick / TailleTick * Margin_Rate Profit : (prix_clôture - prix_ouverture) * Taille_Contrat * Lots
SYMBOL_CALC_MODE_CFDLEVERAGE	Mode Levier CFD - calcul de la marge et du profit pour les CFD avec levier	Marge : (Lots * TailleContrat*PrixMarché) / Levier * Margin_Rate Profit : (prix_clôture - prix_ouverture) * Taille_Contrat * Lots

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
SYMBOL_CALC_MODE_EXCH_STOCKS	Mode boursier - calcul de la marge et du profit pour le trading des actions en bourse	Marge : Lots * TailleContrat*LastPrice * Margin_Rate Profit : (prix_clôture - prix_ouverture) * Taille_Contrat * Lots
SYMBOL_CALC_MODE_EXCH_FUTURES	Mode Futures - calcul de la marge et du profit pour le trading des contrats de futures en bourse	Marge : Lots * MargeInitiale * Margin_Rate ou Lots*MargeMaintien * Margin_Rate Profit : (prix_clôture - prix_ouverture) * Lots * PrixTick / TailleTick
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	Mode Futures	Marge : Lots * MargeInitiale * Margin_Rate ou Lots * MargeMaintien * Margin_Rate

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	FORTS - calcul de la marge et du profit pour les contrats de futures sur FORTS. La marge peut être réduite par le montant de la déviation MarginDiscount selon les règles suivantes : 1. Si le	Profit : (prix_clôture - prix_ouverture) * Lots*PrixTick / TailleTick

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	prix d'une position longue (ordre buy) est inférieur au prix estimé, MarginDiscount = Lots*((PriceSettle-PrixOrdre)*PrixTick/TailleTick) 2. Si le prix d'une position courte (ordre sell) est supérieur au	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	prix estimé, MarginDiscount = Lots*((PrixOrdre-PrixSettle)*PrixTick/TailleTick) avec : ○ PriceSettle - prix estimé (cle	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	aring) de la session précédente; ○ PrixOrdre - prix de la p	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	ositionouprixd'ouvertureemoyenpondérédansl'ordre(	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	demande); ○ PrixTick-prix du tick (coût d'un changement	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	○ Taille de prix d'un point) Taille Tick - taille du tick (pas de ch	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	angement minimum du prix)	
SYMBOL_CALC_MODE_EXCH_BONDS	mode Obligation Boursière - calcul de la marge et du profit pour les obligations en bourse	Marge : Lots * TailleContrat * ValeurFaciale * prix_ouverture * /100 Profit : Lots * prix_clôture * ValeurFaciale * Taille_Contrat + IntérêtsCourus * Lots * Taille_Contrat
SYMBOL_CALC_MODE_EXCH_STOCKS_MOEX	mode	Marge : Lots * TailleContrat * PrixLast * Taux_Marge

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	Actions MOEX - calcul de la marge et du profit pour les actions sur MOEX	Profit : (prix_clôture - prix_ouverture) * Taille_Contrat * Lots
SYMBOL_CALC_MODE_EXCH_BONDS_MOEX	mode Obligations MOEX - calcul de la marge et du profit pour les obligations sur MOEX	Marge : Lots * TailleContrat * ValeurFaciale * prix_ouverture * /100 Profit : Lots * prix_clôture * ValeurFaciale * Taille_Contrat + IntérêtsCourus * Lots * Taille_Contrat
SYMBOL_CALC_MODE_SERV_COLLATERAL	Mode Collatéral - un symb	Marge : non Profit : non Valeur du Marché : Lots * TailleContrat * PrixMarché * TauxLiqudité

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	ole est utilisé comme actif non négociable pour un compte de trading. La valeur de marché d'une position ouverte est calculée sur la base du volume, du prix actuel du marché, de la taille du contrat et	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	du ratio de liquidité. La valeur est incluse dans les Actifs, qui sont ajoutés aux Fonds. Les positions ouvertes de tels symboles augmente le montant de la Marge Libre et sont utilisées comme marge	

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Formule
	supplémentaire (collatéral) pour les positions ouvertes des instruments négociables.	

Il existe plusieurs modes de trading. Les informations sur les modes de trading d'un symbole donné se trouvent dans les valeurs de l'énumeration ENUM_SYMBOL_TRADE_MODE.

ENUM_SYMBOL_TRADE_MODE

Identifiant	Description
SYMBOL_TRADE_MODE_DISABLED	Le trading est désactivé pour le symbole
SYMBOL_TRADE_MODE_LONGONLY	Seules les positions longues sont autorisées
SYMBOL_TRADE_MODE_SHORTONLY	Seules les positions courtes sont autorisées
SYMBOL_TRADE_MODE_CLOSEONLY	Seules les opérations de clôture de position sont autorisées
SYMBOL_TRADE_MODE_FULL	Aucune restriction de trading

Les modes d'exécution possibles d'une transaction pour un symbole donné sont définis dans l'énumeration ENUM_SYMBOL_TRADE_EXECUTION.

ENUM_SYMBOL_TRADE_EXECUTION

Identifiant	Description
SYMBOL_TRADE_EXECUTION_REQUEST	Exécution à la demande

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description
SYMBOL_TRADE_EXECUTION_INSTANT	Exécution instantanée
SYMBOL_TRADE_EXECUTION_MARKET	Exécution au marché
SYMBOL_TRADE_EXECUTION_EXCHANGE	Exécution en bourse

Les méthodes de calcul des swaps lors des transferts de positions sont spécifiées dans l'énumération ENUM_SYMBOL_SWAP_MODE. La méthode de calcul du swap détermine les unités de mesure des paramètres SYMBOL_SWAP_LONG et SYMBOL_SWAP_SHORT. Par exemple, si les swaps sont facturés dans la devise de dépôt du client, alors les valeurs de ces paramètres sont spécifiées comme le montant d'argent dans la devise de dépôt du client.

ENUM_SYMBOL_SWAP_MODE

Identifiant	Description
SYMBOL_SWAP_MODE_DISABLED	Swaps désactivés (aucun swap)
SYMBOL_SWAP_MODE_POINTS	Les swaps sont facturés en points
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Les swaps sont facturés dans la devise de base du symbole
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Les swaps sont facturés dans la devise de marge du symbole
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Les swaps sont facturés dans la devise du dépôt du client
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Les swaps sont facturés en tant qu'intérêt annuel spécifié à partir du prix de l'instrument au moment du calcul du swap (l'année bancaire standard est de 360 jours)
SYMBOL_SWAP_MODE_INTEREST_OPEN	Les swaps sont facturés en tant qu'intérêt annuel spécifié à partir du prix d'ouvertue de la position (l'année bancaire standard est de 360 jours)
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Les swaps sont facturés en réouvrant les positions. A la fin d'une journée de trading, la position est fermée. Le jour suivant, elle est réouverte au prix de clôture +/- le nombre spécifié de points (paramètres SYMBOL_SWAP_LONG et SYMBOL_SWAP_SHORT)
SYMBOL_SWAP_MODE_REOPEN_BID	Les swaps sont facturés en réouvrant les positions. A la fin d'une journée de trading, la position est fermée. Le jour suivant, elle est réouverte au prix Bid courant +/- le nombre spécifié de points (paramètres SYMBOL_SWAP_LONG et SYMBOL_SWAP_SHORT)

© 2000-2024, MetaQuotes Ltd.

Les valeurs de l'énumération ENUM_DAY_OF_WEEK sont utilisées pour spécifier les jours de la semaine.

ENUM_DAY_OF_WEEK

Identifiant	Description
SUNDAY	Dimanche
MONDAY	Lundi
TUESDAY	Mardi
WEDNESDAY	Mercredi
THURSDAY	Jeudi
FRIDAY	Vendredi
SATURDAY	Samedi

Une option est un contrat, qui donne le droit, mais pas l'obligation, d'acheter ou de vendre un actif sous-jacent (biens, actions, futures, etc.) à un prix donné à une date donnée ou avant cette date. Les énumérations suivantes décrivent les propriétés des options, notamment le type de l'option et le droit y afférant.

ENUM_SYMBOL_OPTION_RIGHT

Identifiant	Description
SYMBOL_OPTION_RIGHT_CALL	Une option call vous donne le droit d'acheter un actif à un prix spécifié
SYMBOL_OPTION_RIGHT_PUT	Une option put vous donne le droit de vendre un actif à un prix spécifié

ENUM_SYMBOL_OPTION_MODE

Identifiant	Description
SYMBOL_OPTION_MODE_EUROPEAN	Une option européenne ne peut être exécutée qu'à une date donnée (expiration, date d'exécution, date de livraison)
SYMBOL_OPTION_MODE_AMERICAN	Une option américaine peut être exercée durant n'importe quel jour de trading avant son expiration. La période pendant laquelle un acheteur peut exercer son option est spécifiée dans l'option elle-même

Les instruments financiers sont classés par secteurs de l'économie. Un secteur économique est une partie de l'activité économique qui a des caractéristiques, des objectifs économiques, des fonctions et

© 2000-2024, MetaQuotes Ltd.

des comportements spécifiques, qui permettent de séparer ce secteur des autres parties de l'économie. ENUM_SYMBOL_SECTOR répertorie les secteurs économiques auxquels un instrument de trading peut appartenir.

ENUM_SYMBOL_SECTOR

ID	Description
SECTOR_UNDEFINED	Non défini
SECTOR_BASIC_MATERIALS	Matériaux de base
SECTOR_COMMUNICATION_SERVICES	Services de communication
SECTOR_CONSUMER_CYCLICAL	Consommation cyclique
SECTOR_CONSUMER_DEFENSIVE	Consommateur défensif
SECTOR_CURRENCY	Devises
SECTOR_CURRENCY_CRYPTO	Crypto-monnaies
SECTOR_ENERGY	Énergie
SECTOR_FINANCIAL	Finance
SECTOR_HEALTHCARE	Soins de santé
SECTOR_INDUSTRIALS	Industriels
SECTOR_REAL_ESTATE	Immobilier
SECTOR_TECHNOLOGY	Technologie
SECTOR_UTILITIES	Utilitaires

Chaque instrument financier peut être affecté à un type spécifique d'industrie ou de branche économique. Une industrie est une branche d'une économie qui produit un ensemble étroitement lié de matières premières, de biens ou de services. ENUM_SYMBOL_INDUSTRY répertorie les industries auxquelles un instrument de trading peut appartenir.

ENUM_SYMBOL_INDUSTRY

ID	Description
INDUSTRY_UNDEFINED	Non défini
Matériaux de base	
INDUSTRY_AGRICULTURAL_INPUTS	Intrants agricoles
INDUSTRY_ALUMINIUM	Aluminium
INDUSTRY_BUILDING_MATERIALS	Matériaux de construction
INDUSTRY_CHEMICALS	Produits chimiques

© 2000-2024, MetaQuotes Ltd.

ID	Description
INDUSTRY_COKING_COAL	Charbon à coke
INDUSTRY_COPPER	Cuivre
INDUSTRY_GOLD	Or
INDUSTRY_LUMBER_WOOD	Production de bois d'OEuvre et de bois
INDUSTRY_INDUSTRIAL_METALS	Autres métaux industriels et mines
INDUSTRY_PRECIOUS_METALS	Autres métaux précieux et mines
INDUSTRY_PAPER	Papier et produits en papier
INDUSTRY_SILVER	Argent
INDUSTRY_SPECIALTY_CHEMICALS	Produits chimiques spécialisés
INDUSTRY_STEEL	Acier
Services de communication	
INDUSTRY_ADVERTISING	Agences de publicité
INDUSTRY_BROADCASTING	Diffusion
INDUSTRY_GAMING_MULTIMEDIA	Jeux électroniques et multimédia
INDUSTRY_ENTERTAINMENT	Divertissement
INDUSTRY_INTERNET_CONTENT	Contenu et informations Internet
INDUSTRY_PUBLISHING	Édition
INDUSTRY_TELECOM	Services de télécommunications
Consommation cyclique	
INDUSTRY_APPAREL_MANUFACTURING	Fabrication de vêtements
INDUSTRY_APPAREL_RETAIL	Vente au détail de vêtements
INDUSTRY_AUTO_MANUFACTURERS	Constructeurs automobiles
INDUSTRY_AUTO_PARTS	Pièces automobiles
INDUSTRY_AUTO_DEALERSHIP	Concessionnaires d'automobiles et de camions
INDUSTRY_DEPARTMENT_STORES	Grands magasins
INDUSTRY_FOOTWEAR_ACCESSORIES	Chaussures et accessoires
INDUSTRY_FURNISHINGS	Ameublement, agencements et électroménagers
INDUSTRY_GAMBLING	Jeux d'argent
INDUSTRY_HOME_IMPROV_RETAIL	Vente au détail de rénovation immobilière
INDUSTRY_INTERNET_RETAIL	Vente au détail sur Internet

© 2000-2024, MetaQuotes Ltd.

ID	Description
INDUSTRY_LEISURE	Loisir
INDUSTRY_LODGING	Hébergement
INDUSTRY_LUXURY_GOODS	Produits de luxe
INDUSTRY_PACKAGING_CONTAINERS	Emballage et contenants
INDUSTRY_PERSONAL_SERVICES	Services personnels
INDUSTRY_RECREATIONAL_VEHICLES	Véhicules récréatifs
INDUSTRY_RESIDENT_CONSTRUCTION	Construction résidentielle
INDUSTRY_RESORTS_CASINOS	Resorts et casinos
INDUSTRY_RESTAURANTS	Restaurants
INDUSTRY_SPECIALTY_RETAIL	Commerce de détail spécialisé
INDUSTRY_TEXTILE_MANUFACTURING	Fabrication textile
INDUSTRY_TRAVEL_SERVICES	Services de voyage
Consommateur défensif	
INDUSTRY_BEVERAGES_BREWERS	Boissons - Brasseurs
INDUSTRY_BEVERAGES_NON_ALCO	Boissons - Non alcoolisées
INDUSTRY_BEVERAGES_WINERIES	Boissons - Caves et distilleries
INDUSTRY_CONFECTIONERS	Confiseurs
INDUSTRY_DISCOUNT_STORES	Magasins discount
INDUSTRY_EDUCATION_TRAINIG	Services d'éducation et de formation
INDUSTRY_FARM_PRODUCTS	Produits de la ferme
INDUSTRY_FOOD_DISTRIBUTION	Distribution alimentaire
INDUSTRY_GROCERY_STORES	Épiceries
INDUSTRY_HOUSEHOLD_PRODUCTS	Produits ménagers et personnels
INDUSTRY_PACKAGED_FOODS	Aliments emballés
INDUSTRY_TOBACCO	Tabac
Énergie	
INDUSTRY_OIL_GAS_DRILLING	Forage pétrolier et gazier
INDUSTRY_OIL_GAS_EP	Extraction et traitement du pétrole et du gaz
INDUSTRY_OIL_GAS_EQUIPMENT	Équipements et services pétroliers et gaziers
INDUSTRY_OIL_GAS_INTEGRATED	Pétrole et gaz intégrés

© 2000-2024, MetaQuotes Ltd.

ID	Description
INDUSTRY_OIL_GAS_MIDSTREAM	Pétrole et gaz à mi-chemin
INDUSTRY_OIL_GAS_REFINING	Raffinage et commercialisation du pétrole et du gaz
INDUSTRY_THERMAL_COAL	Charbon thermique
INDUSTRY_URANIUM	Uranium
Finance	
INDUSTRY_EXCHANGE_TRADED_FUND	Fonds négociés en bourse (ETF)
INDUSTRY_ASSETS_MANAGEMENT	Gestion d'actifs
INDUSTRY_BANKS_DIVERSIFIED	Banques - Diversifiées
INDUSTRY_BANKS_REGIONAL	Banques - régionales
INDUSTRY_CAPITAL_MARKETS	Marchés de capitaux
INDUSTRY_CLOSE_END_FUND_DEBT	Fonds à capital fixe - Dette
INDUSTRY_CLOSE_END_FUND_EQUITY	Fonds à capital fixe - Actions
INDUSTRY_CLOSE_END_FUND_FOREIGN	Fonds à capital fixe - Étranger
INDUSTRY_CREDIT_SERVICES	Services de crédit
INDUSTRY_FINANCIAL_CONGLOMERATE	Conglomérats financiers
INDUSTRY_FINANCIAL_DATA_EXCHANGE	Données financières et bourse
INDUSTRY_INSURANCE_BROKERS	Courtiers d'assurance
INDUSTRY_INSURANCE_DIVERSIFIED	Assurance - Diversifiée
INDUSTRY_INSURANCE_LIFE	Assurance - Vie
INDUSTRY_INSURANCE_PROPERTY	Assurances - Propriété et dommages
INDUSTRY_INSURANCE_REINSURANCE	Assurance - Réassurance
INDUSTRY_INSURANCE_SPECIALTY	Assurance - Spécialité
INDUSTRY_MORTGAGE_FINANCE	Financement hypothécaire
INDUSTRY_SHELL_COMPANIES	Sociétés Shell
Soins de santé	
INDUSTRY_BIOTECHNOLOGY	Biotechnologie
INDUSTRY_DIAGNOSTICS_RESEARCH	Diagnostics et recherche
INDUSTRY_DRUGS_MANUFACTURERS	Fabricants de médicaments - généralités
INDUSTRY_DRUGS_MANUFACTURERS_SPEC	Fabricants de médicaments - Spécialités et génériques

© 2000-2024, MetaQuotes Ltd.

ID	Description
INDUSTRY_HEALTHCARE_PLANS	Plans de santé
INDUSTRY_HEALTH_INFORMATION	Services d'information sanitaire
INDUSTRY_MEDICAL_FACILITIES	Installations de soins médicaux
INDUSTRY_MEDICAL_DEVICES	Équipements médicaux
INDUSTRY_MEDICAL_DISTRIBUTION	Distribution médicale
INDUSTRY_MEDICAL_INSTRUMENTS	Instruments et fournitures médicaux
INDUSTRY_PHARM_RETAILERS	Détaillants pharmaceutiques
Industriels	
INDUSTRY_AEROSPACE_DEFENSE	Aérospatiale et défense
INDUSTRY_AIRLINES	Compagnies aériennes
INDUSTRY_AIRPORTS_SERVICES	Aéroports et services aériens
INDUSTRY_BUILDING_PRODUCTS	Produits et équipements de construction
INDUSTRY_BUSINESS_EQUIPMENT	Matériel et fournitures de bureau
INDUSTRY_CONGLOMERATES	Conglomérats
INDUSTRY_CONSULTING_SERVICES	Services de consulting
INDUSTRY_ELECTRICAL_EQUIPMENT	Matériel électrique et pièces
INDUSTRY_ENGINEERING_CONSTRUCTION	Ingénierie et construction
INDUSTRY_FARM_HEAVY_MACHINERY	Machines agricoles et de construction lourde
INDUSTRY_INDUSTRIAL_DISTRIBUTION	Distribution industrielle
INDUSTRY_INFRASTRUCTURE_OPERATIONS	Opérations d'infrastructure
INDUSTRY_FREIGHT_LOGISTICS	Fret et logistique intégrés
INDUSTRY_MARINE_SHIPPING	Expédition maritime
INDUSTRY_METAL_FABRICATION	Fabrication métallique
INDUSTRY_POLLUTION_CONTROL	Contrôle de la pollution et du traitement
INDUSTRY_RAILROADS	Chemins de fer
INDUSTRY_RENTAL_LEASING	Services de location et de crédit-bail
INDUSTRY_SECURITY_PROTECTION	Services de sécurité et de protection
INDUSTRY_SPEALITY_BUSINESS_SERVICES	Services commerciaux spécialisés
INDUSTRY_SPEALITY_MACHINERY	Machines industrielles spécialisées
INDUSTRY_STUFFING_EMPLOYMENT	Services du personnel et d'emploi

© 2000-2024, MetaQuotes Ltd.

ID	Description
INDUSTRY_TOOLS_ACCESSORIES	Outils et accessoires
INDUSTRY_TRUCKING	Camionnage
INDUSTRY_WASTE_MANAGEMENT	Gestion des déchets
Immobilier	
INDUSTRY_REAL_ESTATE_DEVELOPMENT	Développement immobilier
INDUSTRY_REAL_ESTATE_DIVERSIFIED	Immobilier - Diversifié
INDUSTRY_REAL_ESTATE_SERVICES	Services immobiliers
INDUSTRY_REIT_DIVERSIFIED	FPI - Diversifié
INDUSTRY_REIT_HEALTCARE	REIT - Établissements de santé
INDUSTRY_REIT_HOTEL_MOTEL	REIT - Hôtel et motel
INDUSTRY_REIT_INDUSTRIAL	FPI - Industriel
INDUSTRY_REIT_MORTAGE	FPI - Hypothèque
INDUSTRY_REIT_OFFICE	REIT - Bureaux
INDUSTRY_REIT_RESIDENTAL	FPI - Résidentiel
INDUSTRY_REIT_RETAIL	FPI - Commerce de détail
INDUSTRY_REIT_SPECIALITY	FPI - Spécialité
Technologie	
INDUSTRY_COMMUNICATION_EQUIPMENT	Équipement de communication
INDUSTRY_COMPUTER_HARDWARE	Matériel informatique
INDUSTRY_CONSUMER_ELECTRONICS	Electronique grand public
INDUSTRY_ELECTRONIC_COMPONENTS	Composants electroniques
INDUSTRY_ELECTRONIC_DISTRIBUTION	Distribution électronique et informatique
INDUSTRY_IT_SERVICES	Services de technologie de l'information
INDUSTRY_SCIENTIFIC_INSTRUMENTS	Instruments scientifiques et techniques
INDUSTRY_SEMICONDUCTOR_EQUIPMENT	Équipements et matériaux semi-conducteurs
INDUSTRY_SEMICONDUCTORS	Semi-conducteurs
INDUSTRY_SOFTWARE_APPLICATION	Applications logicielles
INDUSTRY_SOFTWARE_INFRASTRUCTURE	Logiciel - Infrastructure
INDUSTRY_SOLAR	Solaire
Utilitaires	

© 2000-2024, MetaQuotes Ltd.

ID	Description
INDUSTRY_UTILITIES_DIVERSIFIED	Services publics - diversifiés
INDUSTRY_UTILITIES_POWERPRODUCERS	Services publics - Producteurs d'électricité indépendants
INDUSTRY_UTILITIES_RENEWABLE	Utilitaires - Renouvelable
INDUSTRY_UTILITIES_REGULATED_ELECTRIC	Utilitaires - Électrique réglementé
INDUSTRY_UTILITIES_REGULATED_GAS	Services publics - Gaz réglementé
INDUSTRY_UTILITIES_REGULATED_WATER	Services publics - Eau réglementée
INDUSTRY_UTILITIES_FIRST	Début de l'énumération des types de services utilitaires. Correspond à INDUSTRY_UTILITIES_DIVERSIFIED.
INDUSTRY_UTILITIES_LAST	Fin de l'énumération des types de services utilitaires. Correspond à INDUSTRY_UTILITIES_REGULATED_WATER.

© 2000-2024, MetaQuotes Ltd.

Propriétés du Compte

Plusieurs fonctions existent pour obtenir des informations sur le compte actuel : AccountInfoInteger(), AccountInfoDouble() et AccountInfoString(). Les paramètres des fonctions sont l'une des valeurs de l'énumération ENUM_ACCOUNT_INFO.

Pour la fonction AccountInfoInteger()

ENUM_ACCOUNT_INFO_INTEGER

Identificateur	Description	Type
ACCOUNT_LOGIN	Numéro du compte	long
ACCOUNT_TRADE_MODE	Mode de trading du compte	ENUM_ACCOUNT_TRADE_MODE
ACCOUNT_LEVERAGE	Levier du compte	long
ACCOUNT_LIMIT_ORDERS	Nombre maximum autorisé d'ordre en attente actifs	int
ACCOUNT_MARGIN_SO_MODE	Mode pour définir la marge minimum autorisée	ENUM_ACCOUNT_STOPOUT_MODE

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
ACCOUNT_TRADE_ALLOWED	Trading autorisé pour le compte actuel	bool
ACCOUNT_TRADE_EXPERT	Trading autorisé pour un Expert Advisor	bool
ACCOUNT_MARGIN_MODE	Mode de calcul de la marge	ENUM_ACCOUNT_MARGIN_MODE
ACCOUNT_CURRENCY_DIGITS	Le nombre de décimales dans la devise du compte, qui sont nécessaires pour un affichage précis des résultats de trading	int
ACCOUNT_FIFO_CLOSE	Indication permet	bool

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	tant de savoir que les positions ne peuvent être fermées que par une règle FIFO. Si la propriété est mise à true, alors chaque position sera fermé dans le même ordre que leur ouverture, en commençant avec la plus ancienne. Dans le cas d'une tentative de fermer les positions dans un ordre	

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	différent, le trader reçevra une erreur correspondante. Pour les comptes avec un mode de comptabilisation autre que le hedging (ACCOUNT_MARGIN_MODE!=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), la valeur de la propriété est toujours false.	

Pour la fonction AccountInfoDouble()

ENUM_ACCOUNT_INFO_DOUBLE

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
ACCOUNT_BALANCE	Solde du compte dans la devise du dépôt	double
ACCOUNT_CREDIT	Crédit du compte dans la devise du dépot	double
ACCOUNT_PROFIT	Profit actuel d'un compte dans la devise du dépôt	double
ACCOUNT_EQUITY	Fonds du comptes dans la devise du dépot	double
ACCOUNT_MARGIN	Marge utilisée du compte dans la devise du dépôt	double
ACCOUNT_MARGIN_FREE	Marge libre d'un	double

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	compte dans la devise du dépôt	
ACCOUNT_MARGIN_LEVEL	Niveau de marge du compte en pourcentage	double
ACCOUNT_MARGIN_SO_CALL	Niveau de l'appel de marge. Suivant le mode, ACCOUNT_MARGIN_SO_MODE est exprimé en pourcentage ou dans la devise du dépôt	double
ACCOUNT_MARGIN_SO_SO	Niveau de marge stop out. Suivant le mode, ACCOUNT_MA	double

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	RGIN_SO_MODE est exprimé en pourcentage ou dans la devise du dépôt	
ACCOUNT_MARGIN_INITIAL	Marge initiale. Le montant réservé sur le compte pour couvrir la marge de tous les ordres en attente	double
ACCOUNT_MARGIN_MAINTENANCE	Marge de maintien. Les fonds minimum réservés sur le compte pour couvrir le	double

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	montant minimum de toutes les positions ouvertes	
ACCOUNT_ASSETS	Les actifs courants d'un compte	double
ACCOUNT_LIABILITIES	Le passif courant d'un compte	double
ACCOUNT_COMMISSION_BLOCKED	Le montant des commissions actuellement bloqué sur un compte	double

Pour la fonction AccountInfoString()

ENUM_ACCOUNT_INFO_STRING

Identificateur	Description	Type
ACCOUNT_NAME	Nom du client	string
ACCOUNT_SERVER	Nom du	string

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	serveur de trades	
ACCOUNT_CURRENCY	Devise du compte	string
ACCOUNT_COMPANY	Nom de la société gérant le compte	string

Plusieurs types de comptes peuvent être ouverts sur un serveur de trades. Le type de compte sur lequel un programme MQL5 est en cours d'exécution peut être trouvé avec l'énumération ENUM_ACCOUNT_TRADE_MODE.

ENUM_ACCOUNT_TRADE_MODE

Identificateur	Description
ACCOUNT_TRADE_MODE_DEMO	Compte de démo
ACCOUNT_TRADE_MODE_CONTEST	Compte de concours
ACCOUNT_TRADE_MODE_REAL	Compte réel

Dans le cas où les fonds ne sont pas suffisants pour maintenir les positions ouvertes, la situation de Stop Out, c'est à dire la clôture forcée, survient. Le niveau minimum de marge auquel le Stop Out se produit peut être défini en pourcentage ou en termes monétaires. Pour connaître le mode défini pour le compte, utilisez l'énumération ENUM_ACCOUNT_STOPOUT_MODE.

ENUM_ACCOUNT_STOPOUT_MODE

Identificateur	Description
ACCOUNT_STOPOUT_MODE_PERCENT	Mode de stop out du compte en pourcentage
ACCOUNT_STOPOUT_MODE_MONEY	Mode de stop out deu compte en valeur

ENUM_ACCOUNT_MARGIN_MODE

Identificateur	Description
ACCOUNT_MARGIN_MODE_RETAIL_NETTING	Utilisé pour les marchés OTC pour interpréter les positions en mode "netting" (une seule position peut exister pour un symbole). La

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description
	marge est calculée sur la base du type du symbole (SYMBOL_TRADE_CALC_MODE).
ACCOUNT_MARGIN_MODE_EXCHANGE	Utilisé pour les marchés boursiers. La marge est calculée sur la base des remises spécifiées dans les paramètres du symbole. Les remises sont définies par le courtier, mais ne peuvent être inférieures aux valeurs définies par la Bourse.
ACCOUNT_MARGIN_MODE_RETAIL_HEDGING	Utilisé pour les marchés boursiers où les positions individuelles sont possibles (hedging, plusieurs positions peuvent exister pour un même symbole). La marge est calculée sur la base du type du symbole (SYMBOL_TRADE_CALC_MODE) prenant en compte la marge hedgée (SYMBOL_MARGIN_HEDGED).

Un exemple de script affichant quelques informations du compte.

```
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- Nom de la société
   string company=AccountInfoString(ACCOUNT_COMPANY);
//--- Nom du client
   string name=AccountInfoString(ACCOUNT_NAME);
//--- Numéro du compte
   long login=AccountInfoInteger(ACCOUNT_LOGIN);
//--- Nom du serveur
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- Devise du compte
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- Compte de démo, de championnat ou réel
   ENUM_ACCOUNT_TRADE_MODE account_type=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(AC
//--- Transforme la valeur de l'énumération sous une forme lisible
   string trade_mode;
   switch(account_type)
     {
      case  ACCOUNT_TRADE_MODE_DEMO:
         trade_mode="demo";
         break;
      case  ACCOUNT_TRADE_MODE_CONTEST:
         trade_mode="championnat";
         break;
      default:
```

© 2000-2024, MetaQuotes Ltd.

```
         trade_mode="réel";
         break;
      }
//--- Le Stop Out est définit en pourcentage ou en monnaie
   ENUM_ACCOUNT_STOPOUT_MODE stop_out_mode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoInteg
//--- Récupère la valeur des niveaux lorsque le Margin Call et le Stop Out se produise
   double margin_call=AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL);
   double stop_out=AccountInfoDouble(ACCOUNT_MARGIN_SO_SO);
//--- Affiche des informations sur le compte
   PrintFormat("Le compte du client '%s' #%d %s ouvert en '%s' sur le serveur '%s'",
               name,login,trade_mode,company,server);
   PrintFormat("Devise du compte - %s, les niveaux de MarginCall et de StopOut levels
               currency,(stop_out_mode==ACCOUNT_STOPOUT_MODE_PERCENT)?"pourcentage":"
   PrintFormat("MarginCall=%G, StopOut=%G",margin_call,stop_out);
  }
```

© 2000-2024, MetaQuotes Ltd.

Statistiques de Test

Une fois que le test est terminé, différents paramètres statistiques des résultats de trading sont calculés. Les valeurs des paramètres peuvent être obtenus avec la fonction TesterStatistics(), en spécifiant l'identifiant du paramètre (énumération ENUM_STATISTICS).

Bien que deux types de paramètres (int et double) sont utilisés pour le calcul des statistiques, la fonction retourne toutes les valeurs sous forme de double. Toutes les valeurs statistiques de type double sont exprimées dans la devise du dépôt par défaut, sauf mention contraire.

ENUM_STATISTICS

Identifiant	Description du paramètre statistique	Type
STAT_INITIAL_DEPOSIT	La valeur du dépôt initial	double
STAT_WITHDRAWAL	Retraits du dépôt	double
STAT_PROFIT	Profit net après le test, la somme de STAT_GROSS_PROFIT et STAT_GROSS_LOSS (STAT_GROSS_LOSS est toujours inférieur ou égal à zéro)	double
STAT_GROSS_PROFIT	Profit total, la somme de toutes les positions profitables (positives). La valeur est supérieure ou égale à zéro	double
STAT_GROSS_LOSS	Perte totale, la somme de toutes les positions négatives. La valeur est inférieure ou égale à zéro	double
STAT_MAX_PROFITTRADE	Profit maximum - la plus grande valeur de toutes les positions profitables. La valeur est supérieure ou égale à zéro	double
STAT_MAX_LOSSTRADE	Perte maximum - la plus basse valeur de toutes les positions perdantes. La valeur	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description du paramètre statistique	Type
	est inférieure ou égale à zéro	
STAT_CONPROFITMAX	Profit maximum dans une série de positions profitables. La valeur est supérieure ou égale à zéro	double
STAT_CONPROFITMAX_TRADES	Le nombre de positions ayant formé le STAT_CONPROFITMAX (profit maximum dans une série de positions profitables)	int
STAT_MAX_CONWINS	Le profit total de la plus longue série de positions profitables	double
STAT_MAX_CONPROFIT_TRADES	Le nombre de positions dans la plus longue série de positions profitables STAT_MAX_CONWINS	int
STAT_CONLOSSMAX	Perte maximum dans une série de positions perdantes. La valeur est inférieure ou égale à zéro	double
STAT_CONLOSSMAX_TRADES	Le nombre de positions ayant formé le STAT_CONLOSSMAX (perte maximum dans une série de positions perdantes)	int
STAT_MAX_CONLOSSES	La perte totale de la plus longue série de positions perdantes	double
STAT_MAX_CONLOSS_TRADES	Le nombre de positions dans la plus longue série de positions perdantes STAT_MAX_CONLOSSES	int

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description du paramètre statistique	Type
STAT_BALANCEMIN	Valeur minimum du solde	double
STAT_BALANCE_DD	Chute maximale du solde en termes monéraires. Durant le trading, le solde peut subir de nombreuses chutes, la plus grande valeur est considérée ici	double
STAT_BALANCEDD_PERCENT	La chute du solde en pourcentage, enregistré au moment de la chute maximale du solde en termes monértaires (STAT_BALANCE_DD).	double
STAT_BALANCE_DDREL_PERCENT	Chute maximum du solde en pourcentage. Durant le trading, le solde peut subir plusieurs chutes, pour lesquelles la valeur de la chute relative en pourcentage est calculée. La plus grande valeur est retournée	double
STAT_BALANCE_DD_RELATIVE	Chute du solde en termes monétaires enregistrée au moment de la chute maximale du solde, en pourcentages (STAT_BALANCE_DDREL_PERCENT).	double
STAT_EQUITYMIN	Valeur minimum des fonds	double
STAT_EQUITY_DD	Chute maximum des fonds en termes monétaires. Durant le trading, plusieurs chutes peuvent apparaître sur les	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description du paramètre statistique	Type
	fonds ; la plus grande valeur apparaît ici.	
STAT_EQUITYDD_PERCENT	Chute en pourcentage, enregistrée au moment de la chute maximale des fonds, en termes monétaires (STAT_EQUITY_DD).	double
STAT_EQUITY_DDREL_PERCENT	Chute maximum des fonds en pourcentage. Durant le trading, les fonds peuvent subir plusieurs chutes, pour lesquelles la valeur de la chute relative en pourcentage est calculée. La plus grande valeur est retournée	double
STAT_EQUITY_DD_RELATIVE	Chute des fonds en termes monétaires enregistrée au moment de la chute maximale des fonds, en pourcentage (STAT_EQUITY_DDREL_PERCENT).	double
STAT_EXPECTED_PAYOFF	Rendement attendu	double
STAT_PROFIT_FACTOR	Facture de profit, égal au ratio de STAT_GROSS_PROFIT/STAT_GROSS_LOSS. Si STAT_GROSS_LOSS=0, le facture de profit est égal à DBL_MAX	double
STAT_RECOVERY_FACTOR	Facteur de recouvrement, égal au ratio de STAT_PROFIT/STAT_BALANCE_DD	double
STAT_SHARPE_RATIO	Ratio de Sharpe	double

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description du paramètre statistique	Type
STAT_MIN_MARGINLEVEL	Valeur minimum du niveau de marge	double
STAT_CUSTOM_ONTESTER	La valeur du critère d'optimisation personnalisé calculé et retourné par la fonction OnTester()	double
STAT_DEALS	Le nombre de transactions	int
STAT_TRADES	Le nombre de positions	int
STAT_PROFIT_TRADES	Positions profitables	int
STAT_LOSS_TRADES	Positions perdantes	int
STAT_SHORT_TRADES	Positions courtes	int
STAT_LONG_TRADES	Positions longues	int
STAT_PROFIT_SHORTTRADES	Positions courtes gagnantes	int
STAT_PROFIT_LONGTRADES	Positions longues gagnantes	int
STAT_PROFITTRADES_AVGCON	Longueur moyenne d'une série gagnante de trades	int
STAT_LOSSTRADES_AVGCON	Longueur moyenne d'une série perdante de trades	int
STAT_COMPLEX_CRITERION	Critère d'optimisation complexe	

© 2000-2024, MetaQuotes Ltd.

Constantes de Trading

Diverses constantes utilisées pour la programmation des stratégies de trading sont divisées en groupes suivants :

- Propriétés de la Base de Données de l'Historique – reçoit les informations générales du symbole ;
- Propriétés de l'ordre – pour obtenir des informations sur les ordres ;
- Propriétés de la position – pour obtenir des informations sur les positions actuelles ;
- Propriétés de la transaction – pour obtenir des informations sur les transactions ;
- Types d'opération de trading – description des opérations de trading disponibles ;
- Types des transactions de trading - description des types de transaction de trading possibles ;
- Ordres de trading en DOM – séparation des ordres selon la direction de l'opération demandée.

© 2000-2024, MetaQuotes Ltd.

Propriétés de la Base de Données de l'Historique

Lors de l'accès aux timeseries, la fonction SeriesInfoInteger() est utilisée pour otenir des informations sur le symbole supplémentaires. L'identifiant d'une propriété requise est passé comme paramètre de la fonction. L'identifiant peut être une des valeurs de ENUM_SERIES_INFO_INTEGER.

ENUM_SERIES_INFO_INTEGER

Identificateur	Description	Type
SERIES_BARS_COUNT	Le nombre de barres pour le symbole et la période actuels	long
SERIES_FIRSTDATE	La toute première date pour le symbole et la période actuellement	datetime
SERIES_LASTBAR_DATE	Heure d'ouverture de la dernière barre du symbole et de la période	datetime
SERIES_SERVER_FIRSTDATE	La toute première date dans l'historique du symbole sur le serveur quelle que soit la période	datetime
SERIES_TERMINAL_FIRSTDATE	La toute première date dans l'historique du symbole sur le terminal client quelle que soit la période	datetime
SERIES_SYNCHRONIZED	Indicateur de synchronisation du symbole et de la période actuels	bool

© 2000-2024, MetaQuotes Ltd.

Propriétés des Ordres

Les demandes d'exécution des trades sont formalisées comme des ordres. Chaque ordre a une variété de propriétés pour la lecture. Les informations les concernant peuvent être obtenues en utilisant les fonctionsOrderGet...() etHistoryOrderGet...().

Pour les fonctions OrderGetInteger() et HistoryOrderGetInteger()

ENUM_ORDER_PROPERTY_INTEGER

Identificateur	Description	Type
ORDER_TICKET	Ticket de l'ordre. Numéro unique attribué à chaque ordre	long
ORDER_TIME_SETUP	Heure d'installation de l'ordre	datetime
ORDER_TYPE	Type d'ordre	ENUM_ORDER_TYPE
ORDER_STATE	Etat de l'ordre	ENUM_ORDER_STATE
ORDER_TIME_EXPIRATION	Heure d'expiration de l'ordre	datetime
ORDER_TIME_DONE	Heure d'éxécution et d'annulation de l'ordre	datetime
ORDER_TIME_SETUP_MSC	L'heure de placement d'un ordre pour son exécution en millisecondes depuis le 01.01.1970	long
ORDER_TIME_DONE_MSC	Heure d'éxécution/annulation d'un ordre en millisecondes depuis le 01.01.1970	long
ORDER_TYPE_FILLING	Type de remplissage	ENUM_ORDER_TYPE_FILLING

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	de l'ordre	
ORDER_TYPE_TIME	Durée de l'ordre	ENUM_ORDER_TYPE_TIME
ORDER_MAGIC	ID d'un Expert Advisor qui a passé l'ordre (conçu pour s'assurer que chaque Expert Advisor place son propre numéro unique)	long
ORDER_REASON	La raison ou la source du placement d'un ordre	ENUM_ORDER_REASON
ORDER_POSITION_ID	L'identifiant de la position qui définit un ordre dés qu'il est éxécuté. Chaque ordre exécuté donne lieu à une transaction qui ouvre ou modifie une position déjà existante. L'identifiant de cette position est définit à l'ordre exécuté à ce moment.	long
ORDER_POSITION_BY_ID	L'identifiant d'une position opposée utilisé pour fermer par l'ordre	long

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	ORDER_TYPE _CLOSE_BY	

Pour les fonctions OrderGetDouble() et HistoryOrderGetDouble()

ENUM_ORDER_PROPERTY_DOUBLE

Identificateur	Description	Type
ORDER_VOLUME_INITIAL	Volume de l'ordre initial	double
ORDER_VOLUME_CURRENT	Volume de l'ordre courant	double
ORDER_PRICE_OPEN	Prix indiqué de l'ordre	double
ORDER_SL	Valeur du Stop Loss	double
ORDER_TP	Valeur du Take Profit	double
ORDER_PRICE_CURRENT	Le prix actuel de l'ordre	double
ORDER_PRICE_STOPLIMIT	Le prix de l'ordre Limit pour l'ordre StopLimit	double

Pour les fonctions OrderGetString() et HistoryOrderGetString()

ENUM_ORDER_PROPERTY_STRING

Identificateur	Description	Type
ORDER_SYMBOL	Symbole de l'ordre	string
ORDER_COMMENT	Commentaire de l'ordre	string
ORDER_EXTERNAL_ID	Identifiant de l'ordre dans un système de trading externe (sur l'Echange)	string

© 2000-2024, MetaQuotes Ltd.

Lors de l'envoi d'une demande de trade avec la fonction OrderSend(), certaines opérations nécessite d'indiquer le type de l'ordre. Le type de l'ordre est spécifié dans le champ *type* de la structure spéciale MqlTradeRequest, et peut être l'une des valeurs de l'énumération ENUM_ORDER_TYPE.

ENUM_ORDER_TYPE

Identificateur	Description
ORDER_TYPE_BUY	Ordre d'Achat au Marché
ORDER_TYPE_SELL	Ordre de Vente au Marché
ORDER_TYPE_BUY_LIMIT	Ordre en attente Buy Limit
ORDER_TYPE_SELL_LIMIT	Ordre en attente Sell Limit
ORDER_TYPE_BUY_STOP	Ordre en attente Buy Stop
ORDER_TYPE_SELL_STOP	Ordre en attente Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	Une fois le prix de l'ordre atteint, un ordre en attente Buy Limit est placé au prix StopLimit
ORDER_TYPE_SELL_STOP_LIMIT	Une fois le prix de l'ordre atteint, un ordre en attente Sell Limit est placé au prix StopLimit
ORDER_TYPE_CLOSE_BY	Ordre pour fermer une position par une position opposée

Chaque ordre a un statut qui décrit son état. Pour obtenir une information, utilisez OrderGetInteger() ou HistoryOrderGetInteger() avec le modificateur ORDER_STATE. Les valeurs autorisées sont stockées dans l'énumération ENUM_ORDER_STATE.

ENUM_ORDER_STATE

Identificateur	Description
ORDER_STATE_STARTED	Ordre vérifié, mais pas encore accepté par le courtier
ORDER_STATE_PLACED	Ordre accepté
ORDER_STATE_CANCELED	Ordre annulé par le client
ORDER_STATE_PARTIAL	Ordre partiellement éxécuté
ORDER_STATE_FILLED	Ordre complètement éxécuté
ORDER_STATE_REJECTED	Ordre rejeté
ORDER_STATE_EXPIRED	Ordre expiré
ORDER_STATE_REQUEST_ADD	L'ordre est enregistré (placé dans le système de trading)
ORDER_STATE_REQUEST_MODIFY	L'ordre est modifié (modification de ses paramètres)
ORDER_STATE_REQUEST_CANCEL	L'ordre est annulé (annulé dans le système de trading)

© 2000-2024, MetaQuotes Ltd.

Lors de l'envoi d'un ordre pour exécution **à l'heure actuelle** (heure en vigueur), le prix et le volume d'achat/vente requis doivent être spécifiés. Gardez également à l'esprit que les marchés financiers ne garantissent pas que l'intégralité du volume demandé soit disponible pour un certain instrument financier au prix souhaité. Par conséquent, les opérations de trading en temps réel sont régulées en utilisant les modes d'exécution prix et volume. Les modes, ou politiques d'exécution, définissent les règles pour les cas où le prix a changé ou le volume demandé ne peut pas être entièrement satisfait **pour le moment**.

Le mode d'exécution des prix peut être obtenu à partir de la propriété SYMBOL_TRADE_EXEMODE du symbole, contenant la combinaison d'indicateurs de l'énumération ENUM_SYMBOL_TRADE_EXECUTION.

Mode d'exécution	Description	La valeur dans ENUM_SYMBOL_TRADE_EXECUTION
Mode d'exécution (Demande de l'Exécution)	Exécute un ordre au marché au prix précédemment reçu du courtier. Les prix pour un ordre au marché sont demandés au courtier avant l'envoi de l'ordre. A la réception des prix, l'exécution de l'ordre au prix donné peut	SYMBOL_TRADE_EXECUTION_REQUEST

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	La valeur dans ENUM_SYMBOL_TRADE_EXECUTION
	être soit confirmé, soit rejeté.	
Exécution Instantanée (Exécution Instantanée)	Exécution immédiate d'un ordre au marché au prix spécifié. Lors de l'envoi de la demande d'un ordre à exécuter, la plateforme ajoute automatiquement les prix actuels à l'ordre. • Si le courtier	SYMBOL_TRADE_EXECUTION_INSTANT

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	La valeur dans ENUM_SYMBOL_TRADE_EXECUTION
	• racceptelepri x, l'ordreestexécuté. • Silecourtiern'ac	

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	La valeur dans ENUM_SYMBOL_TRADE_EXECUTION
	c e p t e p a s l e p r i x d e m a n d é , u n e " r e c o t a t i o n " e s t e n v o y é e :	

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	La valeur dans ENUM_SYMBOL_TRADE_EXECUTION
	le courtier renvoie les prix auxquels ses ordres peuvent être	

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	La valeur dans ENUM_SYMBOL_TRADE_EXECUTION
	exécuté.	
Exécution au Marché (Exécution au Marché)	Le courtier choisit le prix d'exécution de l'ordre sans autre discussion avec le trader. L'envoi de l'ordre dans ce mode vaut consentement préalable à son exécution à ce prix.	SYMBOL_TRADE_EXECUTION_MARKET
Exécution Boursière (Exécution Boursière)	Les opérations de trading sont exécutées aux prix des	SYMBOL_TRADE_EXECUTION_EXCHANGE

© 2000-2024, MetaQuotes Ltd.

Mode d'exécution	Description	La valeur dans ENUM_SYMBOL_TRADE_EXECUTION
	offres de marché en cours.	

La politique de remplissage du volume est spécifiée dans la propriété ORDER_TYPE_FILLING de l'ordre et ne peut contenir que des valeurs de l'énumération ENUM_ORDER_TYPE_FILLING

Politique de remplissage	Description	Une valeur de ENUM_ORDER_TYPE_FILLING
Rempli ou Annulé (Fill or Kill)	Un ordre ne peut être exécuté qu'avec le volume spécifié. Si le volume nécessaire d'un instrument financier est actuellement indisponible sur le marché, l'ordre ne sera pas exécuté.	ORDER_FILLING_FOK

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	Description	Une valeur de ENUM_ORDER_TYPE_FILLING
	Le volume souhaité peut être composé de plusieurs offres disponibles. La possibilité d'utiliser des ordres FOK est déterminée sur le serveur de trading.	
Immédiat ou Annulé (Immediate or Cancel)	Un trader s'engage à exécuter une transaction avec le volume disponible maximum sur le marché dans la limite de	ORDER_FILLING_IOC

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	Description	Une valeur de ENUM_ORDER_TYPE_FILLING
	celui indiqué dans l'ordre. Si la demande ne peut pas être satisfaite complètement, un ordre avec le volume disponible sera exécuté et le volume restant sera annulé. La possibilité d'utiliser les ordres IOC est déterminée sur le serveur de trading.	
Passif (Book or Cancel)	L'ordre BoC	ORDER_FILLING_BOC

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	Description	Une valeur de ENUM_ORDER_TYPE_FILLING
	suppose que l'ordre ne peut être placé que dans le Depth of Market et ne peut pas être exécuté immédiatement. Si l'ordre peut être exécuté immédiatement après avoir été passé, il est alors annulé. En fait, la politique BOC garantit que le prix de l'ordre passé	

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	Description	Une valeur de ENUM_ORDER_TYPE_FILLING
	sera moins bon que le prix actuel. Les ordres BoC sont utilisés pour mettre en œuvre le trading passif, de sorte que l'ordre n'est pas exécuté immédiatement lorsqu'il est placé et n'affecte pas la liquidité actuelle. Seuls les ordres limit et stop limit sont	

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	Description	Une valeur de ENUM_ORDER_TYPE_FILLING
	supportés (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT, ORDER_TYPE_SELL_STOP_LIMIT).	
Retour	En cas de remplissage partiel, un ordre avec un volume restant n'est pas annulé mais traité ultérieurement. Les ordres Retour ne sont pas	ORDER_FILLING_RETURN

© 2000-2024, MetaQuotes Ltd.

Politique de remplissage	Description	Une valeur de ENUM_ORDER_TYPE_FILLING
	autorisés en mode d'Exécution au Marché (exécution au marché — SYMBOL_TRADE_EXECUTION_MARKET).	

Lors de l'envoi d'une demande de trade à l'aide de la fonction OrderSend(), la politique d'exécution du volume nécessaire peut être définie dans *type_filling*, à savoir dans la structure spéciale MqlTradeRequest. Les valeurs de l'énumération ENUM_ORDER_TYPE_FILLING sont disponibles. Pour obtenir la valeur de la propriété dans un ordre actif/terminé spécifique, utilisez les fonctions OrderGetInteger() ou HistoryOrderGetInteger() avec le modificateur ORDER_TYPE_FILLING.

Avant d'envoyer un ordre avec l'heure d'exécution actuelle, pour le réglage correct de la valeur ORDER_TYPE_FILLING (type d'exécution du volume), vous pouvez utiliser la fonction SymbolInfoInteger() avec chaque instrument financier pour obtenir la valeur de la propriété SYMBOL_FILLING_MODE, qui montre les types d'exécution de volume autorisé pour le symbole sous la forme d'une combinaison de drapeaux. Le type de remplissage ORDER_FILLING_RETURN est activé à tout moment sauf pour le mode "Exécution au marché" (SYMBOL_TRADE_EXECUTION_MARKET).

L'utilisation des types de remplissage en fonction du mode d'exécution peut être représentée dans le tableau suivant :

Type d'Exécution\Politique de Remplissage	Rempli ou Annulé / Fill or Kill (FOK ORDER_FILLING_FOK)	Immédiat ou Annulé / Immediate or Cancel (IOC ORDER_FILLING_IOC)	Retour (ORDER_FILLING_RETURN)
Exécution Instantanée (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (quel que soit les paramètres du symbole)	+ (quel que soit les paramètres du symbole)	+ (toujours)
Demande de l'Exécution SYMBOL_TRADE_EXECUTION_REQUEST	+ (quel que soit les paramètres du symbole)	+ (quel que soit les paramètres du symbole)	+ (toujours)

© 2000-2024, MetaQuotes Ltd.

Type d'Exécution\Politique de Remplissage	Rempli ou Annulé / Fill or Kill (FOK ORDER_FILLING_FOK)	Immédiat ou Annulé / Immediate or Cancel (IOC ORDER_FILLING_IOC)	Retour (ORDER_FILLING_RETURN)
Exécution au Marché SYMBOL_TRADE_EXECUTION_MARKET	+ (défini dans les paramètres du symbole)	+ (défini dans les paramètres du symbole)	- (désactivé quel que soit les paramètres du symbole)
Exécution Boursière SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (défini dans les paramètres du symbole)	+ (défini dans les paramètres du symbole)	+ (toujours)

En cas d'ordres en attente, le type de remplissage ORDER_FILLING_RETURN doit être utilisé quel que soit le type d'exécution (SYMBOL_TRADE_EXEMODE), car ces ordres ne sont pas destinés à être exécutés au moment de l'envoi. Lors de l'utilisation d'ordres en attente, le trader accepte à l'avance que, lorsque les conditions d'un accord sur cet ordre sont remplies, le courtier utilisera le type de remplissage pris en charge par la place boursière.

La période de validité de l'ordre peut être définie dans le champ *type_time* de la structure MqlTradeRequest lors de l'envoi d'une demande de trade en utilisant la fonction OrderSend(). L'énumération des valeurs de ENUM_ORDER_TYPE_TIME est autorisée. Pour obtenir la valeur de cette propriété, utilisez la fonction OrderGetInteger() ou HistoryOrderGetInteger() avec le modificateur ORDER_TYPE_TIME.

ENUM_ORDER_TYPE_TIME

Identificateur	Description
ORDER_TIME_GTC	Valide jusqu'à l'annulation de l'ordre
ORDER_TIME_DAY	Valide pour la journée courante de trading
ORDER_TIME_SPECIFIED	Valide jusqu'à l'expiration spécifiée
ORDER_TIME_SPECIFIED_DAY	L'ordre sera effectif jusqu'à 23:59:59 du jour spécifié. Si cette heure est en dehors d'une séance de trading, l'ordre expire dans l'heure de trading le plus proche.

La raison du placement de l'ordre est contenue dans la propriété ORDER_REASON. Un ordre peut être placé par un programme MQL5, depuis une application mobile, comme le résultat d'un StopOut, etc. Les valeurs possibles de ORDER_REASON sont décrites dans l'énumération ENUM_ORDER_REASON .

ENUM_ORDER_REASON

Identificateur	Description
ORDER_REASON_CLIENT	L'ordre a été placé depuis le terminal de bureau

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description
ORDER_REASON_MOBILE	L'ordre a été placé depuis une application mobile
ORDER_REASON_WEB	L'ordre a été placé depuis une plateforme web
ORDER_REASON_EXPERT	L'ordre a été placé depuis un programme MQL5, c'est à dire par un Expert Advisor ou un script
ORDER_REASON_SL	L'ordre a été placé comme étant le résultat de l'activation d'un Stop Loss
ORDER_REASON_TP	L'ordre a été placé comme étant le résultat de l'activation d'un Take Profit
ORDER_REASON_SO	L'ordre a été placé comme étant le résultat d'un évènement Stop Out

© 2000-2024, MetaQuotes Ltd.

Propriétés d'une Position

L'exécution d'opérations de trading résulte en l'ouverture d'une position, le changement de son volume et/ou de sa direction, ou sa suppression. Les opérations de trading sont conduites sur la base d'ordres, envoyés par la fonction OrderSend() sous la forme de demandes de trade. Pour chaque instrument financier (symbole), il ne peut y avoir qu'une seule position ouverte. Une position a un ensemble de propriétés disponibles en lecture avec les fonctions PositionGet...().

Pour la fonction PositionGetInteger()

ENUM_POSITION_PROPERTY_INTEGER

Identificateur	Description	Type
POSITION_TICKET	Ticket de la position. Numéro unique attribué à chaque position nouvellement ouverte. Il correspond généralement au ticket d'un ordre utilisé pour ouvrir la position, sauf lorsque le ticket est modifié à la suite des opérations de service sur le serveur, par exemple, lors de la facturation des swaps avec la ré-ouverture de la position. Pour trouver une ordre utilisé pour ouvrir une position, appliquez la propriété POSITION_IDENTIFIER. POSITION_TICKET valeur qui correspond à MqlTradeRequest::position.	long

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
POSITION_TIME	Heure d'ouverture d'une position	datetime
POSITION_TIME_MSC	Heure d'ouverture d'une position en millisecondes depuis le 01.01.1970	long
POSITION_TIME_UPDATE	Heure de changement d'une position	datetime
POSITION_TIME_UPDATE_MSC	Heure de changement d'une position en millisecondes depuis le 01.01.1970	long
POSITION_TYPE	Type de position	ENUM_POSITION_TYPE
POSITION_MAGIC	Nombre magique d'une position (voir ORDER_MAGIC)	long
POSITION_IDENTIFIER	L'identifiant de la position est un numéro unique attribué à chaque position ré-ouverte. Il ne change pas tout au long de son cycle de vie et correspond au ticket d'un ordre utilisé pour ouvrir une position. L'identifiant de la position est spécifié dans chaque ordre (ORDER_POSITION_ID) et transaction (DEAL_POSITION	long

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
	_ID) utilisé pour l'ouvrir, le modifier, ou le fermer. Utilisez cette propriété pour rechercher des ordres et des transactions liées à la position. Lors de l'inversion d'une position en mode compensation (netting - n'utilisant qu'un seul trade), POSITION_IDENTIFIER ne change pas. Cependant, POSITION_TICKET est remplacé par le ticket de l'ordre qui a conduit à l'inversion. L'inversion de position n'est pas fourni en mode hedging.	
POSITION_REASON	La raison de l'ouverture d'une position	ENUM_POSITION_REASON

Pour la fonction PositionGetDouble()

ENUM_POSITION_PROPERTY_DOUBLE

Identificateur	Description	Type
POSITION_VOLUME	Volume d'une position	double
POSITION_PRICE_OPEN	Prix d'ouverture d'une position	double
POSITION_SL	Niveau du Stop Loss d'une position ouverte	double

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
POSITION_TP	Niveau du Take Profit d'une position ouverte	double
POSITION_PRICE_CURRENT	Prix actuel du symbole de la position	double
POSITION_SWAP	Swap cumulatif	double
POSITION_PROFIT	Profit actuel	double

Pour la fonction PositionGetString()

ENUM_POSITION_PROPERTY_STRING

Identificateur	Description	Type
POSITION_SYMBOL	Symbole de la position	string
POSITION_COMMENT	Commentaire de la position	string
POSITION_EXTERNAL_ID	Identifiant de la position dans un système externe de trading (en Bourse)	string

La direction d'une position ouverte (achat ou vente) est définie par la valeur de l'énumération ENUM_POSITION_TYPE. Pour obtenir le type d'une position ouverte, utilisez la fonctionPositionGetInteger() avec le modificateur POSITION_TYPE.

ENUM_POSITION_TYPE

Identificateur	Description
POSITION_TYPE_BUY	Achat (Buy)
POSITION_TYPE_SELL	Vente (Sell)

La raison de l'ouverture d'une position est contenue dans la propriété POSITION_REASON. Une position peut être ouverte comme étant le résultat de l'activation d'un ordre placé depuis un terminal de bureau, depuis une application mobile, par un Expert Advisor, etc. Les valeurs possibles de POSITION_REASON sont décrites dans l'énumération ENUM_POSITION_REASON.

ENUM_POSITION_REASON

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description
POSITION_REASON_CLIENT	La position a été ouverte par l'activation d'un ordre placé depuis un terminal de bureau
POSITION_REASON_MOBILE	La position a été ouverte par l'activation d'un ordre placé depuis une application mobile
POSITION_REASON_WEB	La position a été ouverte par l'activation d'un ordre placé depuis une plateforme web
POSITION_REASON_EXPERT	La position a été ouverte par l'activation d'un ordre placé depuis un programme MQL5, c'est à dire un Expert Advisor ou un script

© 2000-2024, MetaQuotes Ltd.

Propriétés des Transactions

Une transaction est le reflet d'un trade exécuté sur la base d'unordre qui contient une demande de trade. Chaque trade est décrit par des propriétés qui permettent d'obtenir des informations le concernant. Pour lire les valeurs des propriétés, on utlise des fonctions de type HistoryDealGet...(), qui renvoient les valeurs des énumérations correspondantes.

Pour la fonction HistoryDealGetInteger()

ENUM_DEAL_PROPERTY_INTEGER

Identificateur	Description	Type
DEAL_TICKET	Ticket de la transaction. Numéro unique attribué à chaque transaction	long
DEAL_ORDER	Transaction numéro de l'ordre	long
DEAL_TIME	Heure de la transaction	datetime
DEAL_TIME_MSC	L'heure de l'éxécution de la transaction en millisecondes depuis le 01.01.1970	long
DEAL_TYPE	Type de transaction	ENUM_DEAL_TYPE
DEAL_ENTRY	Entrée de la transaction - entrée, sortie, renversement	ENUM_DEAL_ENTRY
DEAL_MAGIC	Nombre magique de la transaction (regardez ORDER_MAGIC)	long
DEAL_REASON	La raison ou la source de l'exécution d'une transaction	ENUM_DEAL_REASON
DEAL_POSITION_ID	Identifiant de la position, à l'ouverture, en modification ou à la fermeture de la transaction. Chaque position possède un identifiant unique qui et affecté à toutes les transactions éxécutées pour le symbole pendant toute la durée de vie de la position.	long

Pour la fonction HistoryDealGetDouble()

ENUM_DEAL_PROPERTY_DOUBLE

Identificateur	Description	Type
DEAL_VOLUME	Volume de la transaction	double
DEAL_PRICE	Prix de la transaction	double
DEAL_COMMISSION	Commission de la transaction	double
DEAL_SWAP	Swap cumulatif à la fermeture	double
DEAL_PROFIT	Pofit de la transaction	double

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Type
DEAL_FEE	Frais pour qu'une transaction soit facturée immédiatement après son exécution	double
DEAL_SL	Niveau de Stop Loss • Les transactions d'entrée et d'inversion utilisent les valeurs du Stop Loss de l'ordre d'origine sur la base desquelles la position a été ouverte ou inversée • Les transactions de sortie utilisent le Stop Loss d'une position au moment de la clôture de la position	double
DEAL_TP	Niveau du Take Profit • Les transactions d'entrée et de renversement utilisent les valeurs du Take Profit de l'ordre d'origine sur la base duquel la position a été ouverte ou renversée • Les transactions de sortie utilisent la valeur du Take Profit d'une position au moment de la clôture de la position	double

Pour la fonction HistoryDealGetString()

ENUM_DEAL_PROPERTY_STRING

Identificateur	Description	Type
DEAL_SYMBOL	Symbole de la transaction	string
DEAL_COMMENT	Commentaire de la transaction	string
DEAL_EXTERNAL_ID	L'identifiant de la transaction dans un système de trading externe (sur l'Echange)	string

Chaque transaction est caractérisé par le type, les valeurs autorisées sont énumérées dans ENUM_DEAL_TYPE. Pour obtenir des informations sur le type de transaction, utilisez la fonction HistoryDealGetInteger() avec le modificateur DEAL_TYPE.

ENUM_DEAL_TYPE

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description
DEAL_TYPE_BUY	Achat (Buy)
DEAL_TYPE_SELL	Vente (Sell)
DEAL_TYPE_BALANCE	Solde
DEAL_TYPE_CREDIT	Crédit
DEAL_TYPE_CHARGE	Frais supplémentaires
DEAL_TYPE_CORRECTION	Correction
DEAL_TYPE_BONUS	Bonus
DEAL_TYPE_COMMISSION	Commission supplémentaire
DEAL_TYPE_COMMISSION_DAILY	Commission journalière
DEAL_TYPE_COMMISSION_MONTHLY	Commission mensuelle
DEAL_TYPE_COMMISSION_AGENT_DAILY	Commission d'agent journalière
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Commission d'agent mensuelle
DEAL_TYPE_INTEREST	Taux d'intérêt
DEAL_TYPE_BUY_CANCELED	Transaction d'achat annulée. Il peut y avoir une situation où une transaction d'achat éxécutée précédemment est annulée. Dans ce cas, le type de transaction précédemment éxécutée (DEAL_TYPE_BUY) est changé en DEAL_TYPE_BUY_CANCELED, et son profit/perte est remis à zéro. Le profit/perte précédemment obtenu est facturé/ retiré à l'aide d'une balance séparée
DEAL_TYPE_SELL_CANCELED	Transaction de vente annulée. Il peut y avoir une situation où une transaction de vente éxécutée précédemment est annulée. Dans ce cas, le type de transaction précédemment éxécutée (DEAL_TYPE_SELL) est changé en DEAL_TYPE_SELL_CANCELED, et son profit/perte est remis à zéro. Le profit/perte précédemment obtenu est facturé/ retiré à l'aide d'une balance séparée
DEAL_DIVIDEND	Opérations de dividendes
DEAL_DIVIDEND_FRANKED	Opérations de dividendes franche (non imposable)
DEAL_TAX	Frais d'impôts

Les offres diffèrent non seulement dans leurs types définis dans ENUM_DEAL_TYPE, mais aussi dans la façon dont ils changent de position. Il peut s'agir d'une ouverture de position simple ou d'une accumulation d'une position préalablement ouverte (entrée au marché), d'une fermeture de position

© 2000-2024, MetaQuotes Ltd.

par une transaction opposée d'un volume correspondant (sortie au marché) ou d'une inversion de position si la transaction en sens inverse couvre le volume de la position précédemment ouverte.

Toutes ces situations sont décrites par des valeurs de l'énumération ENUM_DEAL_ENTRY. Pour recevoir ces informations sur une transaction, utilisez la fonction HistoryDealGetInteger() avec le modificateur DEAL_ENTRY.

ENUM_DEAL_ENTRY

Identificateur	Description
DEAL_ENTRY_IN	Entrée
DEAL_ENTRY_OUT	Sortie
DEAL_ENTRY_INOUT	Renversement
DEAL_ENTRY_OUT_BY	Fermer une position par une position opposée

La raison de l'exécution d'une transaction est contenue dans la propriété DEAL_REASON. Une transaction peut être exécutée comme étant le résultat du déclenchement d'un ordre placé depuis une application mobile ou un programme MQL5, mais aussi comme étant le résultat d'un évènement StopOut, du calcul de la marge de variation, etc. Les valeurs possibles de DEAL_REASON sont décrites dans l'énumération ENUM_DEAL_REASON. Pour les transactions hors trading résultant d'opérations liées au solde, au crédit, aux commissions et autres, DEAL_REASON_CLIENT est indiqué comme étant la raison.

ENUM_DEAL_REASON

Identificateur	Description
DEAL_REASON_CLIENT	La transaction a été exécutée comme étant le résultat de l'activation d'un ordre placé depuis un terminal de bureau
DEAL_REASON_MOBILE	La transaction a été exécutée comme étant le résultat de l'activation d'un ordre placé depuis une application mobile
DEAL_REASON_WEB	La transaction a été exécutée comme étant le résultat de l'activation d'un ordre placé depuis la plateforme web
DEAL_REASON_EXPERT	La transaction a été exécutée comme étant le résultat de l'activation d'un ordre placé depuis un programme MQL5, c'est à dire un Expert Advisor ou un script
DEAL_REASON_SL	La transaction a été exécutée comme étant le résultat de l'activation d'un Stop Loss
DEAL_REASON_TP	La transaction a été exécutée comme étant le résultat de l'activation d'un Take Profit
DEAL_REASON_SO	La transaction a été exécutée comme étant le résultat de l'activation d'un évènement Stop Out

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description
DEAL_REASON_ROLLOVER	La transaction a été exécutée en raison d'un rollover
DEAL_REASON_VMARGIN	La transaction a été exécutée après la facturation de la marge de variation
DEAL_REASON_SPLIT	La transaction a été exécutée après le découpage (réduction du prix) d'un instrument sur lequel une position était ouverte au moment de l'annonce du découpage

© 2000-2024, MetaQuotes Ltd.

Types d'Opérations de Trading

Le trading est effectué en envoyant des ordres pour ouvrir des positions avec la fonction OrderSend(), mais aussi pour placer, modifier ou supprimer des ordres en attente. Chaque ordre fait référence au type de l'opération demandée. Les opérations de trading sont décrites dans l'énumération ENUM_TRADE_REQUEST_ACTIONS.

ENUM_TRADE_REQUEST_ACTIONS

Identificateur	Description
TRADE_ACTION_DEAL	Place un trade pour une exécution immédiate avec les paramètres spécifiés (ordre au marché)
TRADE_ACTION_PENDING	Place un trade pour une exécution dans des conditions spécifiées (ordre en attente)
TRADE_ACTION_SLTP	Modifie les valeurs du Stop Loss et du Take Profit d'une position ouverte
TRADE_ACTION_MODIFY	Modifie les paramètres d'un ordre placé précedemment
TRADE_ACTION_REMOVE	Supprime l'ordre en attente placé précedemment
TRADE_ACTION_CLOSE_BY	Ferme une position par une position opposée

Exemple de l'opération de trading TRADE_ACTION_DEAL pour ouvrir une position Buy :

```
#define EXPERT_MAGIC 123456   // MagicNumber de l'
//+------------------------------------------------------------------+
//| Ouverture d'une position Buy                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trading et le résultat de la demande
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- paramètres de la demande
   request.action   =TRADE_ACTION_DEAL;                    // type de l'opération de
   request.symbol   =Symbol();                             // symbole
   request.volume   =0.1;                                  // volume de 0.1 lot
   request.type     =ORDER_TYPE_BUY;                       // type de l'ordre
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // prix d'ouverture
   request.deviation=5;                                    // déviation du prix auto
   request.magic    =EXPERT_MAGIC;                         // MagicNumber de l'ordre
//--- envoi la demande
   if(!OrderSend(request,result))
      PrintFormat("OrderSend erreur %d",GetLastError());     // en cas d'erreur d'envo
//--- informations de l'opération
   PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,result.deal
  }
//+------------------------------------------------------------------+
```

Exemple de l'opération de trading TRADE_ACTION_DEAL pour ouvrir une position Sell :

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber de l'
//+------------------------------------------------------------------+
//| Ouverture d'une position Sell                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trading et le résultat de la demande
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- paramètres de la demande
   request.action   =TRADE_ACTION_DEAL;                     // type de l'opération de
   request.symbol   =Symbol();                              // symbole
   request.volume   =0.1;                                   // volume de 0.2 lot
   request.type     =ORDER_TYPE_SELL;                        // type de l'ordre
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // prix d'ouverture
   request.deviation=5;                                     // déviation du prix auto
   request.magic    =EXPERT_MAGIC;                          // MagicNumber de l'ordre
//--- envoi la demande
   if(!OrderSend(request,result))
      PrintFormat("OrderSend erreur %d",GetLastError());     // en cas d'erreur d'envo
//--- informations de l'opération
   PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,result.deal
  }
<b88>//+------------------------------------------------------------------+
```

Exemple de l'opération de trading TRADE_ACTION_DEAL pour fermer des positions :

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber de l'expert
//+------------------------------------------------------------------+
//| Fermeture de toutes les positions                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trading et le résultat
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // nombre de positions ouvertes
//--- boucle sur toutes les positions ouvertes
   for(int i=total-1; i>=0; i--)
     {
      //--- paramètres de l'ordre
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- affiche les informations de la position
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- si le MagicNumber correspond
      if(magic==EXPERT_MAGIC)
        {
         //--- remise à zéro de la demande et du résultat
         ZeroMemory(request);
         ZeroMemory(result);
         //--- paramètres de l'opration
         request.action   =TRADE_ACTION_DEAL;        // type de l'opération de tradin
         request.position =position_ticket;          // ticket de la position
         request.symbol   =position_symbol;          // symbole
         request.volume   =volume;                   // volume de la position
         request.deviation=5;                        // déviation du prix autorisée
         request.magic    =EXPERT_MAGIC;             // MagicNumber de la position
         //--- définit le prix et le type de l'ordre suivant le type de la position
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
           }
         //--- affiche les informations de clôture
         PrintFormat("Ferme #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- envoi la demande
         if(!OrderSend(request,result))
            PrintFormat("OrderSend erreur %d",GetLastError());  // en cas d'échec de
         //--- informations sur l'opération
         PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,resu
         //---
        }
```

```
      }
   }
//+------------------------------------------------------------------+
```

Exemple de l'opération de trading TRADE_ACTION_PENDING pour placer un ordre en attente :

© 2000-2024, MetaQuotes Ltd.

```
#property description "Exemple de placement d'ordres en attente"
#property script_show_inputs
#define EXPERT_MAGIC 123456                              // MagicNumber de l'expert
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;   // type de l'ordre
//+------------------------------------------------------------------+
//| Placement d'ordres en attente                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trade et le résultat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- paramètres pour placer un ordre en attente
   request.action   =TRADE_ACTION_PENDING;                          // type de trac
   request.symbol   =Symbol();                                      // symbole
   request.volume   =0.1;                                           // volume de 0.
   request.deviation=2;                                             // déviation au
   request.magic    =EXPERT_MAGIC;                                  // MagicNumber
   int offset = 50;                                                 // décalage du
   double price;                                                    // prix de décl
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);             // valeur du po
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);             // nombre de dé
   //--- vérification du type de l'opération
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                      // type de l'or
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;    // prix d'ouver
      request.price    =NormalizeDouble(price,digits);             // prix d'ouver
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                      // type de l'o
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;    // prix d'ouve
      request.price    =NormalizeDouble(price,digits);              // prix d'ouve
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                           // type de l'o
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point; // prix d'ouve
      request.price=NormalizeDouble(price,digits);                 // prix d'ouve
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                      // type de l'o
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;    // prix d'ouve
      request.price    =NormalizeDouble(price,digits);             // prix d'ouve
     }
   else Alert("Cet exemple ne permet de ne placer que des ordres en attente");   // si
//--- envoie la demande
   if(!OrderSend(request,result))
      PrintFormat("OrderSend erreur %d",GetLastError());            // en cas d'é
//--- informations sur l'opération
   PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,result.deal
  }
//+------------------------------------------------------------------+
```

Exemple de l'opération de trading TRADE_ACTION_SLTP pour modifier les valeurs des Stop Loss et Take Profit d'une position ouverte :

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber de l' expert
//+------------------------------------------------------------------+
//| Modification du Stop Loss et du Take Profit d'une position       |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trade et le résultat
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // nombre de positions ouvertes
//--- boucle sur toutes les positions ouvertes
   for(int i=0; i<total; i++)
     {
      //--- paramètres de l'ordre
      ulong  position_ticket=PositionGetTicket(i);// ticket de la position
      string position_symbol=PositionGetString(POSITION_SYMBOL); // symbole
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // nombre d
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // MagicNumber de la position
      double volume=PositionGetDouble(POSITION_VOLUME);    // volume de la position
      double sl=PositionGetDouble(POSITION_SL);  // Stop Loss de la position
      double tp=PositionGetDouble(POSITION_TP);  // Take Profit de la position
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- affiche quelques informations sur la position
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- si le MagicNumber correspond et que le Stop Loss et le Take Profit ne sont
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2024, MetaQuotes Ltd.

```
         //--- calcule les niveaux de prix actuels
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_I
         double price_level;
         //--- si la distance de décalage minimum autorisée en points du prix de clotu
         if(stop_level<=0)
            stop_level=150; // définit la distance de décalage de 150 points du priwx
         else
            stop_level+=50; // définit la distance de décalage à (SYMBOL_TRADE_STOPS_I

         //--- calcul et arrondi des valeurs du Stop Loss et du Take Profit
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(ask+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(bid-price_level,digits);
           }
         //--- remise à zéro de la demande et du résultat
         ZeroMemory(request);
         ZeroMemory(result);
         //--- définition des paramètres de l'opération
         request.action  =TRADE_ACTION_SLTP; // type de l'opération de trading
         request.position=position_ticket;   // ticket de la position
         request.symbol=position_symbol;     // symbole
         request.sl      =sl;                // Stop Loss de la position
         request.tp      =tp;                // Take Profit de la position
         request.magic=EXPERT_MAGIC;         // MagicNumber de la position
         //--- affiche des informations sur la modification
         PrintFormat("Modification de #%I64d %s %s",position_ticket,position_symbol,Er
         //--- envoi de la demande
         if(!OrderSend(request,result))
            PrintFormat("OrderSend erreur %d",GetLastError());  // en cas d'échec de l
         //--- informations sur l'opération
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
      }
   }
//+------------------------------------------------------------------+
```

Exemple de l'opération de trading TRADE_ACTION_MODIFY pour modifier les niveaux de prix d'ordres en attente :

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber de l'expert
//+------------------------------------------------------------------+
//| Modification des ordres en attente                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trade et le résultat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // nombre total d'ordres en attente placés
//--- boucle sur tous les ordres en attente placés
   for(int i=0; i<total; i++)
     {
      //--- paramètres pour l'ordre
      ulong  order_ticket=OrderGetTicket(i);                          // ticket de
      string order_symbol=Symbol();                                   // symbole
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS);  // nombre de
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                      // MagicNumbe
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);             // volume act
      double sl=OrderGetDouble(ORDER_SL);                             // Stop Loss
      double tp=OrderGetDouble(ORDER_TP);                             // Take Profi
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // type de l'
      int offset = 50;                                                // décalage d
      double price;                                                   // prix de dé
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);       // valeur du
      //--- affiche des informations sur l'ordre
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  order_ticket,
                  order_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- si le MagicNumber correspond, et que le Stop Loss et le Take Profit ne son
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                        // type de l'op
         request.order = OrderGetTicket(i);                         // ticket de l'
         request.symbol   =Symbol();                                // symbole
         request.deviation=5;                                       // déviation au
        //--- définition du niveau de prix, du Take Profit et du Stop Loss de l'ordre
         if(type==ORDER_TYPE_BUY_LIMIT)
           {
            price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prix d'
           }
         else if(type==ORDER_TYPE_SELL_LIMIT)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prix d
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
```

```
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prix c
           }
         else if(type==ORDER_TYPE_SELL_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prix c
           }
         //--- envoi de la demande
         if(!OrderSend(request,result))
            PrintFormat("OrderSend erreur %d",GetLastError());  // en cas d'échec de l
         //--- informations sur l'opération
         PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,resul
         //--- remise à zéro de la demande et du résultat
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

Exemple de l'opération de trading TRADE_ACTION_REMOVE pour supprimer des ordres en attente :

```
#define EXPERT_MAGIC 123456  // MagicNumber de l'expert
//+------------------------------------------------------------------+
//| Suppression des ordres en attente                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trading et le résultat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // nombre total d'ordre en attente placés
//--- boucle sur tous les ordres en attente placés
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                  // Ticket de l'ordre
      ulong  magic=OrderGetInteger(ORDER_MAGIC);              // MagicNumber de l'orc
      //--- si le MagicNumber correspond
      if(magic==EXPERT_MAGIC)
        {
         //--- remise à zéro de la demande et du résultat
         ZeroMemory(request);
         ZeroMemory(result);
         //--- paramètres de l'opération
         request.action=TRADE_ACTION_REMOVE;                  // type de l'opération
         request.order = order_ticket;                        // Ticket de l'ordre
         //--- envoi la demande
         if(!OrderSend(request,result))
            PrintFormat("OrderSend erreur %d",GetLastError());  // en cas d'échec de l
         //--- informations sur l'opération
         PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,resul
        }
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

Exemple de trade TRADE_ACTION_CLOSE_BY pour des positions fermées par des positions opposées :

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber de l' expert
//+------------------------------------------------------------------+
//| Ferme toutes les positions par des positions opposées            |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trade et son résultat
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // nombre de positions ouvertes
//--- boucle sur toutes les positions ouvertes
   for(int i=total-1; i>=0; i--)
     {
      //--- paramètres de l'ordre
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      double sl=PositionGetDouble(POSITION_SL);
      double tp=PositionGetDouble(POSITION_TP);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- affiche les informations de la position
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- si le MagicNumber correspond
      if(magic==EXPERT_MAGIC)
        {
         for(int j=0; j<i; j++)
           {
            string symbol=PositionGetSymbol(j); // symbole de la position opposée
            //--- si les symboles de la position initiale et de la position opposée co
            if(symbol==position_symbol && PositionGetInteger(POSITION_MAGIC)==EXPERT_M
              {
               //--- définit le type de la position opposée
               ENUM_POSITION_TYPE type_by=(ENUM_POSITION_TYPE)PositionGetInteger(POSIT
               //--- sortie si les types de la position initiale et de la position opp
               if(type==type_by)
                  continue;
               //--- remise à zéro de la demande et de son résultat
               ZeroMemory(request);
               ZeroMemory(result);
               //--- définition des paramètres de l'opération
               request.action=TRADE_ACTION_CLOSE_BY;                        // type d
               request.position=position_ticket;                            // ticket
               request.position_by=PositionGetInteger(POSITION_TICKET);     // ticket
               //request.symbol     =position_symbol;
               request.magic=EXPERT_MAGIC;                                  // MagicN
               //--- affiche des informations sur la cloture par une position opposée
               PrintFormat("Close #%I64d %s %s by #%I64d",position_ticket,position_sym
               //--- envoie la demande
               if(!OrderSend(request,result))
                  PrintFormat("OrderSend erreur %d",GetLastError()); // en cas d'impos
```

```
                //--- information sur l'opération
                PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result
               }
            }
         }
      }
   }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

Types de Transaction de Trading

Lors de l'exécution de certains actions définies sur un compte de trading, son état change. De telles actions incluent :

- L'envoi d'une demande de trading depuis n'importe application MQL5 dans le terminal client avec les fonctions OrderSend et OrderSendAsync et son exécution ;
- L'envoi d'une demande de trading vie l'interface graphique du terminal et son exécution ;
- L'activation des ordres en attente et des stops sur le serveur ;
- Effectuer des opérations du côté du serveur de trades.

Les transactions de trading suivantes sont effectuées suite à ces actions :

- traitement d'une demande de trading ;
- changement des ordres ouverts ;
- changement de l'historique des ordres ;
- changement de l'historique des transactions ;
- changement des positions.

Par exemple, lors de l'envoi d'un ordre d'achat, il est pris en compte par le gestionnaire, un ordre d'achat correspondant est créé pour le compte, l'ordre est ensuite exécuté et supprimé de la liste des ordres ouverts, puis il est ajouté à l'historique des ordres, une transaction correspondante est ajoutée à l'hisorique et une nouvelle position est créée. Toutes ces actions sont des transactions de trading.

Pour laisser un programmeur suivre les actions effectuées en relation avec un compte de trading, la fonction OnTradeTransaction est fournie. Ce gestionnaire permet d'obtenir les transactions appliquées à un compte dans l'application MQL5. La description de la transaction est soumis dans le premier paramètre de OnTradeTransaction en utilisant la structure MqlTradeTransaction.

Le type de transaction de trading est soumis dans le paramètre type de la structure MqlTradeTransaction. Les différents types de transactions sont décrits dans l'énumération suivante :

ENUM_TRADE_TRANSACTION_TYPE

Identificateur	Description
TRADE_TRANSACTION_ORDER_ADD	Ajout d'un nouvel ordre d'ouverture.
TRADE_TRANSACTION_ORDER_UPDATE	Mise à jour d'un ordre d'ouverture. Les mises à jour incluent non seulement les changements évidents du terminal client ou du serveur de trades, mais également les changements d'état d'un ordre lors de sa définition (par exemple, transition de ORDER_STATE_STARTED à ORDER_STATE_PLACED ou de ORDER_STATE_PLACED à ORDER_STATE_PARTIAL, etc.).
TRADE_TRANSACTION_ORDER_DELETE	Suppression d'un ordre de la liste des ordres ouverts. Un ordre peut être supprimé par des ordres ouverts suite à une requête correspondante ou à son exécution (remplissage) et au déplacement vers l'historique.

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description
TRADE_TRANSACTION_DEAL_ADD	Ajout d'une transaction dans l'historique. L'action est exécutée à la suite de l'exécution d'un ordre ou d'opérations sur le solde de compte.
TRADE_TRANSACTION_DEAL_UPDATE	Mise à jour d'une transaction dans l'historique. Il peut y avoir des cas où la transaction précédemment éxécutée est modifiée sur le serveur. Par exemple, une transaction a été modifiée dans un système de trading externe (bourse) où elle a été précédemment transférée par un courtier.
TRADE_TRANSACTION_DEAL_DELETE	Suppression d'une transaction de l'historique. Il peut y avoir des cas où la transaction précédemment exécutée est supprimée sur le serveur. Par exemple, une transaction a été supprimée dans un système de trading externe (bourse) où elle a été précédemment transférée par un courtier.
TRADE_TRANSACTION_HISTORY_ADD	Ajout d'un ordre à l'historique à la suite de son exécution ou de son annulation.
TRADE_TRANSACTION_HISTORY_UPDATE	Changement d'un ordre situé dans l'historique des ordres. Ce type est fourni pour améliorer la fonctionnalité du côté du serveur de trades.
TRADE_TRANSACTION_HISTORY_DELETE	Suppression d'un ordre de l'historique des ordres. Ce type est fourni pour améliorer la fonctionnalité du côté du serveur de trades.
TRADE_TRANSACTION_POSITION	Modification d'une position non liée à l'exécution d'une transaction. Ce type de transaction montre qu'une position a été modifiée sur le serveur de trades. Le volume de la position, le prix d'ouverture, les niveaux Stop Loss et Take Profit peuvent être modifiés. Les données du changement sons soumises dans une structure MqlTradeTransaction via le gestionnaire OnTradeTransaction. La changement de la position (ajout, changement, ou clôture), comme étant le résultat de l'éxécution d'une transaction, ne mène pas à une transaction TRADE_TRANSACTION_POSITION.
TRADE_TRANSACTION_REQUEST	Notification du fait qu'une demande de trade a été traitée par le serveur et que le résultat du traitement a été reçu. Un seul champ (type de la transaction de trading) doit être analysé pour ces transactions dans la structure MqlTradeTransaction. Les second et troisième paramètres de la fonction OnTradeTransaction (demande et résultat) doivent être analysés pour obtenir des données supplémentaires.

© 2000-2024, MetaQuotes Ltd.

Suivant le type de la transaction de trading, différents paramètres sont remplis dans la structure MqlTradeTransaction la décrivant. Une description détaillée des données soumises est affichée dans "Structure d'une Transaction de Trade".

Voir aussi

Structure de Transaction de Trade , OnTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Ordres de Trading dans le Depth Of Market

Pour les actions, la fenêtre du Depth of Market est disponible, dans laquelle vous pouvez voir les ordres Buy et Sell actuels. La direction souhaitée d'un trade, le montant requis et le prix demandé sont indiqués pour chaque ordre.

Pour obtenir des informations sur l'état actuel du DOM au moyen de MQL5, il faut utiliser la fonction MarketBookGet(), qui place une "copie d'écran" du DOM dans le tableau de structures MqlBookInfo. Le champ *type* de chaque élément du tableau contient des informations sur la direction de l'ordre - une valeur de l'énumération ENUM_BOOK_TYPE.

ENUM_BOOK_TYPE

Identificateur	Description
BOOK_TYPE_SELL	Ordre de vente (Offre)
BOOK_TYPE_BUY	Ordre d'Achat (Demande)
BOOK_TYPE_SELL_MARKET	Ordre de vente au Marché
BOOK_TYPE_BUY_MARKET	Ordre d'achat au Marché

Voir aussi

Structures et classes, Structure du DOM, Types d'opération de trading, Market Info

© 2000-2024, MetaQuotes Ltd.

Propriétés du Signal

Les énumérations suivantes sont utilisées lorsque vous utilisez des signaux de trading et des paramètres de copie de signal.

L'énumération des propriétés de type double du signal de trading :

ENUM_SIGNAL_BASE_DOUBLE

Identifiant	Description
SIGNAL_BASE_BALANCE	Balance du compte
SIGNAL_BASE_EQUITY	Fonds du compte
SIGNAL_BASE_GAIN	Bénéfice du compte
SIGNAL_BASE_MAX_DRAWDOWN	Chute maximale du compte
SIGNAL_BASE_PRICE	Prix de l'abonnement du signal
SIGNAL_BASE_ROI	Retour sur Investissement (%)

L'énumération des propriétés de type integer du signal de trading :

ENUM_SIGNAL_BASE_INTEGER

Identifiant	Description
SIGNAL_BASE_DATE_PUBLISHED	Date de publication (date de disponibilité pour la souscription)
SIGNAL_BASE_DATE_STARTED	Date de début du monitoring
SIGNAL_BASE_DATE_UPDATED	La date de la dernière mise à jour des statistiques de trading du signal
SIGNAL_BASE_ID	ID du Signal
SIGNAL_BASE_LEVERAGE	Levier du compte
SIGNAL_BASE_PIPS	Profit en pips
SIGNAL_BASE_RATING	Taux de la position
SIGNAL_BASE_SUBSCRIBERS	Nombre d'abonnés
SIGNAL_BASE_TRADES	Nombre de trades
SIGNAL_BASE_TRADE_MODE	Type de compte (0-réel, 1-demo, 2-championnat)

L'énumération des propriétés de type string du signal de trading :

ENUM_SIGNAL_BASE_STRING

Identifiant	Description
SIGNAL_BASE_AUTHOR_LOGIN	Login de l'auteur

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description
SIGNAL_BASE_BROKER	Nom du courtier (compagnie)
SIGNAL_BASE_BROKER_SERVER	Serveur du courtier
SIGNAL_BASE_NAME	Nom du signal
SIGNAL_BASE_CURRENCY	Devise du signal de base

L'énumération des propriétés de type double des paramètres de copie de signaux :

ENUM_SIGNAL_INFO_DOUBLE

Identifiant	Description
SIGNAL_INFO_EQUITY_LIMIT	Limite d'Equity
SIGNAL_INFO_SLIPPAGE	Slippage (utilisé lors du placement d'ordres au marché dans la synchronisation des position et la copie des trades)
SIGNAL_INFO_VOLUME_PERCENT	Pourcentage maximal du dépôt utilisé (%), r/o

L'énumération des propriétés de type integer des paramètres de copie de signaux :

ENUM_SIGNAL_INFO_INTEGER

Identifiant	Description
SIGNAL_INFO_CONFIRMATIONS_DISABLED	Le drapeau permet la synchronisation sans dialogue de confirmation
SIGNAL_INFO_COPY_SLTP	Copie du drapeau Stop Loss et du drapeau Take Profit
SIGNAL_INFO_DEPOSIT_PERCENT	Pourcentage du dépôt (%)
SIGNAL_INFO_ID	Id du signal, r/o
SIGNAL_INFO_SUBSCRIPTION_ENABLED	Flag d'autorisation por la "Copie des trades par abonnement"
SIGNAL_INFO_TERMS_AGREE	Flag "Est d'accord avec les termes du service des Signaux", r/o

L'énumération des propriétés de type string des paramètres de copie de signaux :

ENUM_SIGNAL_INFO_STRING

Identifiant	Description
SIGNAL_INFO_NAME	Nom du signal, r/o

Voir aussi

Signaux de trading

© 2000-2024, MetaQuotes Ltd.

Constantes Nommées

Toutes les constantes utilisées en MQL5 peuvent être divisées en groupes :

- Substitutions de Macro Prédéfinies – les valeurs sont remplacées pendant la compilation ;
- Constantes mathématiques – les valeurs de certaines expressions mathématiques ;
- Constantes de Types Numériques – some of the simple type restrictions ;
- Codes des Raisons de Non-Initialisation – description des raisons de non-initialisation ;
- Vérification d'un Pointeur d'Objet – énumeration des types de pointeurs retournés par la fonction CheckPointer() ;
- Autres constantes – toutes les autres constantes.

© 2000-2024, MetaQuotes Ltd.

Substitutions de Macro Prédéfinies

Pour simplifier le déboguage et obtenir des informations sur les opérations d'un programme mql5, il existe des constantes spéciales de macro, dont les valeurs sont définies au moment de la compilation. La façon la plus facile d'utiliser ces constantes est d'afficher les valeurs avec la fonction Print(), comme montré dans l'exemple.

Constante	Description
__CPU_ARCHITECTURE__	Nom de l'architecture (ensemble de commandes) pour laquelle le fichier EX5 est compilé
__DATE__	Date de compilation du fichier sans l'heure (les heures, minutes et secondes sont égales à 0)
__DATETIME__	Date et heure de compilation du fichier
__LINE__	Numéro de ligne dans le code source où la macro est située
__FILE__	Nom du fichier compilé actuel
__PATH__	Chemin absolu vers le fichier en cours de compilation
__FUNCTION__	Nom de la fonction dans laquelle la macro est située
__FUNCSIG__	Signature de la fonction dans laquelle la macro est située. La sortie de la description complète des fonctions peut être utile pour identifier les fonctions surchargées
__MQLBUILD__,__MQL5BUILD__	Numéro de build du compilateur
__COUNTER__	Le compilateur remplace la valeur du compteur de 0 à N-1, où N est un nombre d'utilisations dans le code, pour chaque déclaration __COUNTER__ rencontrée. L'ordre __COUNTER__ est garanti lors de la recompilation du code source sans changement. La valeur __COUNTER__ est calculée de la manière suivante : • la valeur initiale du compteur est 0, • après chaque utilisation du compteur, sa valeur est augmentée de 1, • tout d'abord, le compilateur étend toutes les macros et modèles en code source sur site, • un code distinct est créé pour chaque spécialisation de fonction de modèle, • un code séparé est créé pour chaque spécialisation de class /structure templatée, • ensuite, le compilateur passe par le code source obtenu dans l'ordre défini et remplace chaque utilisation __COUNTER__ détectée par la valeur actuelle du compteur. L'exemple ci-dessous montre comment le compilateur gère le code source et remplace toutes les instances de

© 2000-2024, MetaQuotes Ltd.

Constante	Description
	__COUNTER__ qu'il rencontre par des valeurs séquentiellement croissantes.
__RANDOM__	Le compilateur insère une valeur aléatoire ulong pour chaque déclaration __RANDOM__.

Exemple :

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------------------
//| Fonction d'initialisation de l'expert
//+------------------------------------------------------------------------------
void OnInit()
  {
//--- exemple de sortie de l'information de l'initialisation de l'Expert Advisor
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//--- définit l'intervalle entre les évènements du timer
   EventSetTimer(5);
//---
  }
//+------------------------------------------------------------------------------
//| Fonction de dé-initialisation de l'Expert
//+------------------------------------------------------------------------------
void OnDeinit(const int reason)
  {
//--- exemple de sortie de l'information de la dé-initialisation de l'Expert Advisor
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//---
  }
//+------------------------------------------------------------------------------
//| Fonction de tick de l'Expert
//+------------------------------------------------------------------------------
void OnTick()
  {
//--- sortie de l'information à la réception d'un tick
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
   test2();
//---
  }
//+------------------------------------------------------------------------------
//| test1
//+------------------------------------------------------------------------------
void test1(string par)
  {
//--- sortie d'information dans la fonction
```

```
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__," par=",par);
  }
//+------------------------------------------------------------------------------------
//| test2
//+------------------------------------------------------------------------------------
void test2()
  {
//--- sortie d'information dans la fonction
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
  }
//+------------------------------------------------------------------------------------
//| Gestionnaire d'évènement OnTimer
//+------------------------------------------------------------------------------------
void OnTimer()
  {
//---
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
  }
```

L'exemple pour apprendre à travailler avec la macro __COUNTER__

```
// --- crée une macro pour un affichage rapide de l'expression et de sa valeur dans le
#define print(expr) Print(#expr,"=",expr)

// --- définir la macro personnalisée MACRO_COUNTER via la macro prédéfinie __COUNTER_
#define MACRO_COUNTER __COUNTER__

// --- définit la valeur d'entrée de la variable à l'aide de la macro __COUNTER__
input int InpVariable = __COUNTER__;

// --- définir la valeur de la variable globale à l'aide de la macro __COUNTER__ avant
int ExtVariable = __COUNTER__;

//+------------------------------------------------------------------+
//| la fonction renvoie la valeur __COUNTER__                        |
//+------------------------------------------------------------------+
int GlobalFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| la fonction modèle renvoie la valeur __COUNTER__                 |
//+------------------------------------------------------------------+
template<typename T>
int GlobalTemplateFunc(void)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| la structure avec la méthode retournant __COUNTER__              |
//+------------------------------------------------------------------+
struct A
  {
   int               dummy;  // non utilisé

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| la structure du modèle avec la méthode retournant __COUNTER__    |
//+------------------------------------------------------------------+
template<typename T>
struct B
  {
   int               dummy;  // non utilisé

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| la structure avec la méthode template retournant __COUNTER__     |
//+------------------------------------------------------------------+
struct C
  {
   int               dummy;  // non utilisé

   template<typename T>
   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| la fonction # , qui renvoie la valeur __COUNTER__                |
//+------------------------------------------------------------------+
int GlobalFunc2(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Fonction de démarrage du programme de script                     |
//+------------------------------------------------------------------+
void OnStart(void)
  {
// __COUNTER__ dans la macro et les variables
   print(MACRO_COUNTER);
   print(InpVariable);
   print(ExtVariable);

//--- __COUNTER__ dans les fonctions
   print(GlobalFunc());
   print(GlobalFunc());                // la valeur n'est pas changée
   print(GlobalTemplateFunc<int>());
   print(GlobalTemplateFunc<int>());   // la valeur n'est pas changée
   print(GlobalTemplateFunc<double>());// la valeur a été changée
   print(GlobalFunc2());
   print(GlobalFunc2());               // la valeur n'est pas changée

// __COUNTER__ dans la structure
   A a1, a2;
   print(a1.Method());
   print(a2.Method());                 // la valeur n'est pas changée

// __COUNTER__ dans la structure du modèle
   B<int> b1, b2;
   B<double> b3;
   print(b1.Method());
   print(b2.Method());                 // la valeur n'est pas changée
   print(b3.Method());                 // la valeur a changé

// __COUNTER__ dans la structure avec la fonction template
   C c1, c2;
   print(c1.Method<int>());
   print(c1.Method<double>());         // la valeur a changé
   print(c2.Method<int>());            // la même valeur que lors du premier appel à c

// --- regardons à nouveau __COUNTER__ dans la macro et la variable globale
   print(MACRO_COUNTER);  // la valeur a changé
   print(ExtGlobal2);
  }
// --- définit la valeur de la variable globale à l'aide de la macro __COUNTER__ après
int ExtGlobal2 = __COUNTER__;
//+------------------------------------------------------------------+

/* Résultat
   __COUNTER__=3
   InpVariable=0
   ExtVariable=1
   GlobalFunc()=5
```

© 2000-2024, MetaQuotes Ltd.

```
   GlobalFunc()=5
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<double>()=9
   GlobalFunc2()=7
   GlobalFunc2()=7
   a1.Method()=6
   a2.Method()=6
   b1.Method()=10
   b2.Method()=10
   b3.Method()=11
   c1.Method<int>()=12
   c1.Method<double>()=13
   c2.Method<int>()=12
   __COUNTER__=4
   ExtGlobal2=2

*/
```

© 2000-2024, MetaQuotes Ltd.

Constantes Mathématiques

Les constantes spéciales contenant des valeurs sont réservées pour les expressions mathématiques. Ces constantes peuvent être utilisées n'importe où dans le programme au lieu de calculer leurs valeurs avec les fonctions mathématiques.

Constante	Description	Valeur
M_E	e	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.434294481903251827651
M_LN2	ln(2)	0.693147180559945309417
M_LN10	ln(10)	2.30258509299404568402
M_PI	pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.785398163397448309616
M_1_PI	1/pi	0.318309886183790671538
M_2_PI	2/pi	0.636619772367581343076
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.707106781186547524401

Exemple :

```
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//--- affiche les valeurs des constantes
   Print("M_E = ",DoubleToString(M_E,16));
   Print("M_LOG2E = ",DoubleToString(M_LOG2E,16));
   Print("M_LOG10E = ",DoubleToString(M_LOG10E,16));
   Print("M_LN2 = ",DoubleToString(M_LN2,16));
   Print("M_LN10 = ",DoubleToString(M_LN10,16));
   Print("M_PI = ",DoubleToString(M_PI,16));
   Print("M_PI_2 = ",DoubleToString(M_PI_2,16));
   Print("M_PI_4 = ",DoubleToString(M_PI_4,16));
   Print("M_1_PI = ",DoubleToString(M_1_PI,16));
   Print("M_2_PI = ",DoubleToString(M_2_PI,16));
   Print("M_2_SQRTPI = ",DoubleToString(M_2_SQRTPI,16));
   Print("M_SQRT2 = ",DoubleToString(M_SQRT2,16));
   Print("M_SQRT1_2 = ",DoubleToString(M_SQRT1_2,16));
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

© 2000-2024, MetaQuotes Ltd.

Constantes de Types Numériques

Chaque type numérique simple est prévu pour un certain type de tâche et permet d'optimiser l'exécution d'un programme mql5. Pour une meilleure lisibilité du code et un traitement correct des résultats de calcul, des constantes existent permettant de récupérer les informations sur les restrictions sur un certain type de données simples.

Constante	Description	Valeur
CHAR_MIN	Valeur minimale du type char	-128
CHAR_MAX	Valeur maximale du type char	127
UCHAR_MAX	Valeur maximale du type uchar	255
SHORT_MIN	Valeur minimale du type short	-32768
SHORT_MAX	Valeur maximale du type short	32767
USHORT_MAX	Valeur maximale du type ushort	65535
INT_MIN	Valeur minimale du type int	-2147483648
INT_MAX	Valeur maximale du type int	2147483647
UINT_MAX	Valeur maximale du type uint	4294967295
LONG_MIN	Valeur minimale du type long	-9223372036854775808
LONG_MAX	Valeur maximale du type long	9223372036854775807
ULONG_MAX	Valeur maximale du type ulong	18446744073709551615
DBL_MIN	Valeur minimale positive du type double	2.2250738585072014e-308
DBL_MAX	Valeur maximale du type double	1.7976931348623158e+308
DBL_EPSILON	Valeur minimale qui satisfait la condition : 1.0+DBL_EPSILON != 1.0 (pour le type double)	2.2204460492503131e-016
DBL_DIG	Nombre significatif de décimales pour le type double	15
DBL_MANT_DIG	Nombre de bits dans la mantisse pour le type double	53
DBL_MAX_10_EXP	Valeur décimale maximale du degré d'exposant pour le type double	308
DBL_MAX_EXP	Valeur binaire maximale du degré d'exposant pour le type double	1024
DBL_MIN_10_EXP	Valeur décimale minimale du degré d'exposant pour le type double	(-307)
DBL_MIN_EXP	Valeur binaire minimale du degré d'exposant pour le type double	(-1021)

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Valeur
FLT_MIN	Valeur minimale positive du type float	1.175494351e-38
FLT_MAX	Valeur maximale du type float	3.402823466e+38
FLT_EPSILON	Valeur minimale qui satisfait la condition : 1.0+DBL_EPSILON != 1.0 (pour le type float)	1.192092896e-07
FLT_DIG	Nombre significatif de décimales pour le type float	6
FLT_MANT_DIG	Nombre de bits dans la mantisse pour le type float	24
FLT_MAX_10_EXP	Valeur décimale maximale du degré d'exposant pour le type float	38
FLT_MAX_EXP	Valeur binaire maximale du degré d'exposant pour le type float	128
FLT_MIN_10_EXP	Valeur décimale minimale du degré d'exposant pour le type float	-37
FLT_MIN_EXP	Valeur binaire minimale du degré d'exposant pour le type float	(-125)

Exemple :

```
void OnStart()
  {
//--- affiche les valeurs des constantes
   printf("CHAR_MIN = %d",CHAR_MIN);
   printf("CHAR_MAX = %d",CHAR_MAX);
   printf("UCHAR_MAX = %d",UCHAR_MAX);
   printf("SHORT_MIN = %d",SHORT_MIN);
   printf("SHORT_MAX = %d",SHORT_MAX);
   printf("USHORT_MAX = %d",USHORT_MAX);
   printf("INT_MIN = %d",INT_MIN);
   printf("INT_MAX = %d",INT_MAX);
   printf("UINT_MAX = %u",UINT_MAX);
   printf("LONG_MIN = %I64d",LONG_MIN);
   printf("LONG_MAX = %I64d",LONG_MAX);
   printf("ULONG_MAX = %I64u",ULONG_MAX);
   printf("EMPTY_VALUE = %.16e",EMPTY_VALUE);
   printf("DBL_MIN = %.16e",DBL_MIN);
   printf("DBL_MAX = %.16e",DBL_MAX);
   printf("DBL_EPSILON = %.16e",DBL_EPSILON);
   printf("DBL_DIG = %d",DBL_DIG);
   printf("DBL_MANT_DIG = %d",DBL_MANT_DIG);
   printf("DBL_MAX_10_EXP =  %d",DBL_MAX_10_EXP);
   printf("DBL_MAX_EXP = %d",DBL_MAX_EXP);
   printf("DBL_MIN_10_EXP = %d",DBL_MIN_10_EXP);
```

© 2000-2024, MetaQuotes Ltd.

```
   printf("DBL_MIN_EXP = %d",DBL_MIN_EXP);
   printf("FLT_MIN = %.8e",FLT_MIN);
   printf("FLT_MAX = %.8e",FLT_MAX);
   printf("FLT_EPSILON = %.8e",FLT_EPSILON);
/*
   CHAR_MIN = -128
   CHAR_MAX = 127
   UCHAR_MAX = 255
   SHORT_MIN = -32768
   SHORT_MAX = 32767
   USHORT_MAX = 65535
   INT_MIN = -2147483648
   INT_MAX = 2147483647
   UINT_MAX = 4294967295
   LONG_MIN = -9223372036854775808
   LONG_MAX = 9223372036854775807
   ULONG_MAX = 18446744073709551615
   EMPTY_VALUE = 1.7976931348623157e+308
   DBL_MIN = 2.2250738585072014e-308
   DBL_MAX = 1.7976931348623157e+308
   DBL_EPSILON = 2.2204460492503131e-16
   DBL_DIG = 15
   DBL_MANT_DIG = 53
   DBL_MAX_10_EXP =  308
   DBL_MAX_EXP = 1024
   DBL_MIN_10_EXP = -307
   DBL_MIN_EXP = -1021
   FLT_MIN = 1.17549435e-38
   FLT_MAX = 3.40282347e+38
   FLT_EPSILON = 1.19209290e-07
*/
  }
```

© 2000-2024, MetaQuotes Ltd.

Codes des Raisons de Non-Initialisation

Les codes des raisons de non-initialisation sont retournés par la fonction UninitializeReason(). Les valeurs possibles sont les suivantes :

Constante	Valeur	Description
REASON_PROGRAM	0	L'Expert Advisor a terminé son opération en appelant la fonction ExpertRemove()
REASON_REMOVE	1	Le programme a été supprimé du graphique
REASON_RECOMPILE	2	Le programme a été recompilé
REASON_CHARTCHANGE	3	Le symbole de la période du graphique a été modifié
REASON_CHARTCLOSE	4	Le graphique a été fermé
REASON_PARAMETERS	5	Les paramètres d'entrée ont été changés par l'utilisateur
REASON_ACCOUNT	6	Un autre compte a été activé ou une reconnexion au serveur de trades s'est produite en raison de changements dans les paramètres du compte
REASON_TEMPLATE	7	Un nouveau modèle a été appliqué
REASON_INITFAILED	8	Cette valeur signifie que le gestionnaire OnInit() a retourné une valeur différente de 0
REASON_CLOSE	9	Le terminal a été fermé

Le code de la raison de non-initialisation est également passé en paramètre à la fonction OnDeinit(const int reason).

Exemple :

```
//+------------------------------------------------------------------------------------
//| Récupère la description textuelle
//+------------------------------------------------------------------------------------
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Le compte a été changé";break;
      case REASON_CHARTCHANGE:
         text="Le symbole ou la période ont été changés";break;
      case REASON_CHARTCLOSE:
         text="Le graphique a été fermé";break;
      case REASON_PARAMETERS:
         text="Les paramètres d'entrée ont été changés";break;
      case REASON_RECOMPILE:
```

© 2000-2024, MetaQuotes Ltd.

```
         text="Le programme "+__FILE__+" a été recompilé";break;
      case REASON_REMOVE:
         text="Le programme "+__FILE__+" a été enlevé du graphique";break;
      case REASON_TEMPLATE:
         text="Un nouveau modèle a été appliqué au graphique";break;
      default:text="Autre raison";
     }
//---
   return text;
  }
//+------------------------------------------------------------------------------------
//| Fonction de dé-initialisation de l'Expert
//+------------------------------------------------------------------------------------
void OnDeinit(const int reason)
  {
//--- La première façon d'obtenir le code de raison de non-initialisation
   Print(__FUNCTION__,"_code de raison de non-initalisation = ",reason);
    //--- La deuxième façon d'obtenir le code de raison de non-initialisation
   Print(__FUNCTION__,"_UninitReason = ",getUninitReasonText(_UninitReason));
  }
```

© 2000-2024, MetaQuotes Ltd.

Vérification d'un Pointeur d'Objet

La fonction CheckPointer() est utilisée pour vérifier le type du pointeur d'objet. La fonction retourne une valeur de l'énumération ENUM_POINTER_TYPE. Si un pointeur incorrect est utilisé, l'exécution du programme est immédiatement terminée.

Les objets créés avec l'opérateur new() sont de type POINTER_DYNAMIC. L'opérateur delete() peut et ne doit être utilisé que pour ce type de pointeurs.

Tous les autres pointeurs sont de type POINTER_AUTOMATIC, ce signifie que cet objet a été créé automatiquement par l'environnement du programme mql5. Ces objets sont supprimés automatiquement après leur utilisation.

ENUM_POINTER_TYPE

Constante	Description
POINTER_INVALID	Pointeur incorrect
POINTER_DYNAMIC	Pointeur d'un objet créé par l'opérateur new()
POINTER_AUTOMATIC	Pointeur de n'importe quel objet créé automatiquement (sans avoir utilisé new())

Voir aussi

Erreurs d'exécution, Suppression d'Objet, Opérateur delete, CheckPointer

© 2000-2024, MetaQuotes Ltd.

Autres Constantes

La constante CLR_NONE est utilisée pour signifier l'absence de couleur, cela signifie que l'objet graphique ou les séries graphiques d'un indicateur ne seront pas dessinés. Cette constante n'est pas inclue dans la liste des constantes des couleurs Web, mais elle peut être appliquée pour n'importe quel arguement de type color.

La constante INVALID_HANDLE peut être utilisée pour vérifier les handles des fichiers (voir FileOpen() et FileFindFirst()).

Constante	Description	Valeur
CHARTS_MAX	Le nombre maximum possible de graphiques ouverts en même temps dans le terminal	100
clrNONE	Absence de couleur	-1
EMPTY_VALUE	Valeur vide dans le buffer d'un indicateur	DBL_MAX
INVALID_HANDLE	Handle incorrect	-1
IS_DEBUG_MODE	Flag signifiant que le programme mq5 opère en mode debug	différent de zéro en mode debug, sinon zéro
IS_PROFILE_MODE	Flag signifiant que le programme mq5 opère en mode profilage	différent de zéro en mode profilage, sinon zéro
NULL	Zéro pour n'importe quel type	0
WHOLE_ARRAY	Signifie le nombre d'éléments restants jusqu'à la fin du tableau, c'est à dire que le tableau sera traité en entier	-1
WRONG_VALUE	La constante peut être castée implicitement dans n'importe quel type d'énumeration type	-1

La constante EMPTY_VALUE correspond habituellement aux valeurs des indicateurs qui ne sont pas affichés dans le graphique. Par exemple, pour l'indicateur intégré Standard Deviation (Ecart type) avec une période de 20, la ligne des 19 premières barres de l'historique ne sont pas affichées sur le graphique. Si vous créez un handle sur cet indicateur avec la fonction iStdDev() et que vous le copiez vers un tableau des valeurs de l'indicateur pour ces barres avec CopyBuffer(), ces valeurs seront ensuite égales à EMPTY_VALUE.

Vous pouvez choisir de spécifier pour un indicateur personnalisé votre propre valeur vide, lorsque l'indicateur ne devrait pas être dessiné sur le graphique. Utilisez la fonction PlotIndexSetDouble() avec le modificateur PLOT_EMPTY_VALUE.

© 2000-2024, MetaQuotes Ltd.

La constante NULL peut être assignée à une variable de n'importe quel type simple ou à un pointeur vers une structure d'objet ou une classe. L'assignation de NULL à une chaîne de caractères signifie la déinitialisation complète de cette variable.

La constante WRONG_VALUE est prévue pour ces cas, lorsqu'il est nécessaire de retourner la valeur d'une énumeration, et ce doit être une valeur incorrecte. Par exemple, lorsque nous avons besoin d'informer qu'une valeur de retour est une valeur de cette énumération. Considérons comme exemple une fonction CheckLineStyle(), qui retourne le style de ligne d'un objet spécifié par son nom. Si le résultat au moment de la vérification du style par ObjectGetInteger() est true, une valeur de ENUM_LINE_STYLE est retournée ; sinon WRONG_VALUE est retourné.

```
void OnStart()
  {
   if(CheckLineStyle("MyChartObject")==WRONG_VALUE)
      printf("Erreur de récupération du style de ligne.");
  }
//+------------------------------------------------------------------------------
//| Retourne le style de ligne d'un objet spécifié par son nom
//+------------------------------------------------------------------------------
ENUM_LINE_STYLE CheckLineStyle(string name)
  {
   long style;
//---
   if(ObjectGetInteger(0,name,OBJPROP_STYLE,0,style))
      return((ENUM_LINE_STYLE)style);
   else
      return(WRONG_VALUE);
  }
```

La constante WHOLE_ARRAY est prévue pour les fonctions nécessitant de spécifier le nombre d'éléments dans les tableaux traités :

- ArrayCopy();
- ArrayMinimum();
- ArrayMaximum();
- FileReadArray();
- FileWriteArray().

Si vous souhaitez spécifier que toutes les valeurs du tableau, à partir d'une position spécifiée jusqu'à la fin, doivent être traitées, vous devez spécifier la valeur WHOLE_ARRAY.

La constante IS_PROFILE_MODE permet de changer l'opération d'un programme pour la collection de données correcte dans le mode profilage. Le profilage permet de mesurer l'heure d'exécution des fragments individuels d'un programme (comprenant habituellement les fonctions), mais également de mesurer le nombre de ces appels. Les appels à la fonction Sleep() peuvent être désactivé pour déterminer l'heure d'exécution en mode profilage, comme dans cet exemple :

```
//--- Sleep peut grandement affecter les résultat du profilage
if(!IS_PROFILE_MODE) Sleep(100); // désactive l'appel à Sleep() dans le mode profilage
```

© 2000-2024, MetaQuotes Ltd.

La valeur de la constante IS_PROFILE_MODE est définie par le compilateur pendant la compilation, alors qu'elle vaut zéro dans le mode normal. Lors du lancement d'un programme en mode profilage, une compilation spéciale est effectuée et IS_PROFILE_MODE est remplacée avec une valeur différente de zéro.

La constante IS_DEBUG_MODE peut être utile lorsque vous avez besoin de modifier légèrement l'exécution d'un programme mql5 en mode debug. Par exemple, en mode debug vous pouvez avoir besoin d'afficher des informations de déboguage supplémentaires dans le journal du terminal ou de créer des objets graphiques supplémentaires sur le graphique.

L'exemple suivant crée un objet Etiquette et définit sa description et sa couleur suivant le mode d'exécution du script. Afin d'exécuter un script en mode debug depuis MetaEditor, appuyez sur la touche F5. Si vous exécutez le script depuis la fenêtre du navigateur du terminal, alors la couleur et le texte de l'étiquette seront différents.

Exemple :

```
//+------------------------------------------------------------------------------------
//|                                                Check_DEBUG_MODE.mq5
//|                        Copyright © 2009, MetaQuotes Software Corp.
//|                                             https://www.metaquotes.net
//+------------------------------------------------------------------------------------
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------------------------
//| Fonction de démarrage du script
//+------------------------------------------------------------------------------------
void OnStart()
  {
//---
   string label_name="invisible_label";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Objet ",label_name," non trouvé. Code d'erreur = ",GetLastError());
      //--- création de l'étiquette
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- définit la coordonnée X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- définit la coordonnée Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      ResetLastError();
      if(IS_DEBUG_MODE) // mode debug
        {
         //--- affiche le message à propos du mode d'exécution du script
         ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
         //--- définit la couleur du texte à rouge
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
            Print("Impossible de définir la couleur. Erreur",GetLastError());
        }
      else              // mode opérationnel
```

```
        {
         ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
         //--- définit la couleur du texte à invisible
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
            Print("Impossible de définir la couleur. Erreur ",GetLastError());
        }
      ChartRedraw();
      DebugBreak();     // le programme s'arrête ici en mode debug
     }
  }
```

Méthodes de Cryptage

L'énumération ENUM_CRYPT_METHOD est utilisée pour spécifier la méthode de transformation des données, utilisée dans les fonctions CryptEncode() et CryptDecode().

ENUM_CRYPT_METHOD

Constante	Description
CRYPT_BASE64	BASE64
CRYPT_AES128	Cryptage AES avec une clé 128 bits (16 octets)
CRYPT_AES256	Cryptage AES avec une clé 256 bits (32 octets)
CRYPT_DES	Cryptage DES avec une clé 56 bits (7 octets)
CRYPT_HASH_SHA1	Calcul HASH SHA1
CRYPT_HASH_SHA256	Calcul HASH SHA56
CRYPT_HASH_MD5	Calcul HASH MD5
CRYPT_ARCH_ZIP	Archives ZIP

Voir aussi

DebugBreak, Propriétés d'un Programme MQL5 en Cours d'Exécution, CryptEncode(), CryptDecode()

© 2000-2024, MetaQuotes Ltd.

Structures de Données

Le Langage MQL5 offre 12 structures prédéfinies :

- MqlDateTime est conçue pour travailler avec les dates et heures ;
- MqlParam permet d'envoyer des paramètres d'entrée lors de la création d'un gestionnaire de l'indicateur utilisant la fonction IndicatorCreate() ;
- MqlRates est conçue pour manipuler les données historiques, elle contient les informations sur le prix, le volume et le spread ;
- MqlBookInfo est conçue pour obtenir les informations sur le Depth of Market ;
- MqlTradeRequest est utilisée pour créer une demande de trade pour des opérations de trading ;
- MqlTradeCheckResult est conçue pour vérifier la demande de trade préparée avant de l'envoyer ;
- MqlTradeResult contient la réponse du serveur de trades à une demande de trade, envoyée par la fonction OrderSend() ;
- MqlTradeTransaction contient la description d'une transaction de trading ;
- MqlTick est conçue pour récupérer rapidement les informations les plus demandées sur les prix actuels.
- Les structures du calendrier économique sont utilisées pour obtenir les données des évènements du calendrier économique envoyés à la plateforme MetaTrader 5 en temps réel. Les fonctions du calendrier économique permettent d'analyser les paramètres macro-économiques immédiatement après la publication de nouveaux rapports, puisque les valeurs pertinentes sont diffusées directement à partir de la source, sans délai.

© 2000-2024, MetaQuotes Ltd.

MqlDateTime

La structure de type date contient huit champs de type int :

```
struct MqlDateTime
  {
   int year;            // Année
   int mon;             // Mois
   int day;             // Jour
   int hour;            // Heure
   int min;             // Minutes
   int sec;             // Secondes
   int day_of_week;     // Jour de la semaine (0-dimanche, 1-lundi, ... ,6-samedi)
   int day_of_year;     // Numéro du jour dans l'année (0 équivaut au 1er janvier)
  };
```

Note

Le numéro du jour de l'année day_of_year pour une année bissextile, depuis le mois de mars, sera différent du numéro du jour correspondant pour une année bissextile.

Exemple :

```
void OnStart()
  {
//---
   datetime date1=D'2008.03.01';
   datetime date2=D'2009.03.01';

   MqlDateTime str1,str2;
   TimeToStruct(date1,str1);
   TimeToStruct(date2,str2);
   printf("%02d.%02d.%4d, jour de l'année = %d",str1.day,str1.mon,
          str1.year,str1.day_of_year);
   printf("%02d.%02d.%4d, jour de l'année = %d",str2.day,str2.mon,
          str2.year,str2.day_of_year);
  }
/*  Résultat :
  01.03.2008, jour de l'année = 60
  01.03.2009, jour de l'année = 59
*/
```

Voir aussi

TimeToStruct, Structures et Classes

© 2000-2024, MetaQuotes Ltd.

La Structure des Paramètres des Indicateurs (MqlParam)

La structure MqlParam a été spécialement conçue pour fournir des paramètres d'entrée lors de la création du handle d'un indicateur technique avec la fonction IndicatorCreate().

```
struct MqlParam
  {
   ENUM_DATATYPE     type;                    // type du paramètre d'entrée, une valeu
   long              integer_value;           // champ pour stocker un type integer
   double            double_value;            // champ pour stocker un type double
   string            string_value;            // champ pour stocker un type string
  };
```

Tous les paramètres d'entrée d'un indicateur sont transmis sous la forme d'un tableau de type MqlParam, le champ *type* de chaque élément de ce tableau spécifie le type de la donnée transmise par l'élément. Les valeurs de l'indicateur doivent d'abord être placées dans les champs correspondants pour chaque élément (dans *integer_value*, *double_value* ou *string_value*) suivant la valeur de l'énumération ENUM_DATATYPE spécifiée dans le champ *type*.

Si la valeur IND_CUSTOM est passée comme type de l'indicateur à la fonction IndicatorCreate(), le premier élément du tableau des paramètres d'entrée doit avoir le champ *type* avec la valeur de TYPE_STRING de l'énumération ENUM_DATATYPE, et le champ *string_value* doit contenir le nom de l'indicateur personnalisé.

© 2000-2024, MetaQuotes Ltd.

MqlRates

Cette structure stocke les informations des prix, des volumes et du spread.

```
struct MqlRates
  {
   datetime time;          // Début de la période
   double   open;          // Prix d'ouverture
   double   high;          // Le prix le plus haut de la période
   double   low;           // Le prix le plus bas de la période
   double   close;         // Prix de clôture
   long     tick_volume;   // Volume des ticks
   int      spread;        // Spread
   long     real_volume;   // Volume des trades
  };
```

Exemple :

```
void OnStart()
  {
   MqlRates rates[];
   int copied=CopyRates(NULL,0,0,100,rates);
   if(copied<=0)
      Print("Erreur lors de la copie des prix ",GetLastError());
   else Print("Copiés ",ArraySize(rates)," barres");
  }
```

Voir aussi

CopyRates, Accéder aux séries temporelles

© 2000-2024, MetaQuotes Ltd.

MqlBookInfo

Il fournit des informations sur les données du depth of market.

```
struct MqlBookInfo
  {
   ENUM_BOOK_TYPE   type;            // Type d'ordre de l'énumération ENUM_BOOK_TYPE
   double           price;           // Prix
   long             volume;          // Volume
   double           volume_real;     // Volume with greater accuracy
  };
```

Note

La structure du MqlBookInfo est prédéfinie, ainsi elle ne requiert ni de déclaration ni de description. Pour utiliser la structure, il suffit de déclarer une variable.

Le DOM est disponible seulement pour quelques symboles.

Exemple :

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo à propos de ",Symbol());
     }
   else
     {
      Print("Echec de réception du DOM pour le symbole ",Symbol());
     }
```

Voir aussi

MarketBookAdd, MarketBookRelease, MarketBookGet, Ordres de Trading dans le DOM, Types de Données

© 2000-2024, MetaQuotes Ltd.

La Structure de Demande de Trade (MqlTradeRequest)

L'interaction entre le terminal client et le serveur de trades pour l'exécution des opérations de placement des ordres est effectuée en utilisant des demandes de trade. La demande de trade est représentée par une structure prédéfinie de type MqlTradeRequest, laquelle contient tous les champs nécessaires à la réalisation des trades. Le résultat du traitement de la demande est représenté par la structure de type MqlTradeResult.

```
struct MqlTradeRequest
  {
   ENUM_TRADE_REQUEST_ACTIONS    action;          // Type de l'opération de trading
   ulong                         magic;           // Identifiant de l'Expert Advisor
   ulong                         order;           // Ticket de l'ordre
   string                        symbol;          // Symbole de trading
   double                        volume;          // Volume demandé pour une transact
   double                        price;           // Prix
   double                        stoplimit;       // Niveau du StopLimit de l'ordre
   double                        sl;              // Niveau du Stop Loss pour l'ordre
   double                        tp;              // Niveau du Take Profit pour l'ord
   ulong                         deviation;       // Déviation maximale possible pour
   ENUM_ORDER_TYPE               type;            // Type de l'ordre
   ENUM_ORDER_TYPE_FILLING       type_filling;    // Type d'exécution de l'ordre
   ENUM_ORDER_TYPE_TIME          type_time;       // Type d'expiration de l'ordre
   datetime                      expiration;      // Date/heure d'expiration de l'ord
   string                        comment;         // Commentaire de l'ordre
   ulong                         position;        // Ticket de la position
   ulong                         position_by;     // Le ticket d'une position opposée
  };
```

Description des champs

Champ	Description
action	Type d'opération de trade. Peut être l'une des valeurs de l'énumération ENUM_TRADE_REQUEST_ACTIONS.
magic	Identifiant de l'Expert Advisor. Il permet d'organiser le traitement d'analyse des ordres de trading. Chaque Expert Advisor peut générer son propre identifiant unique lors de l'envoi d'une demande de trade.
ordre	Ticket de l'ordre. Il est utilisé pour modifier les ordres en attente.
symbol	Symbole de l'ordre. Il n'est pas nécessaire pour les opérations de modification d'ordre et de fermeture de position.
volume	Volume demandé de l'ordre en lots. Notez que le volume réel d'une transaction dépendra du type d'exécution de l'ordre.
prix	Le prix auquel l'ordre doit être exécuté. Les ordres au marché des symboles dont le type d'exécution est "Exécution au

© 2000-2024, MetaQuotes Ltd.

Champ	Description
	Marché" (SYMBOL_TRADE_EXECUTION_MARKET), de type TRADE_ACTION_DEAL, ne nécessite pas de spécifier un prix.
stoplimit	Le prix auquel l'ordre en attente de type Limit sera placé lorsque le prix atteint la *valeur* prix (cette condition est obligatoire). Jusqu'à ce moment, l'ordre en attente n'est pas placé.
sl	Le prix du Stop Loss dans un mouvement de prix défavorable
tp	Le prix du Take Profit dans un mouvement de prix défavorable
deviation	La déviation maximale du prix, spécifiée en points
type	Type d'ordre. Peut être l'une des valeurs de l'énumération ENUM_ORDER_TYPE.
type_filling	Type d'éxécution de l'ordre. Peut être l'une des valeurs de l'énumération ENUM_ORDER_TYPE_FILLING.
type_time	Type d'expiration de l'ordre. Peut être l'une des valeurs de l'énumération ENUM_ORDER_TYPE_TIME.
expiration	L'heure d'expiration de l'ordre (pour des ordres de type ORDER_TIME_SPECIFIED)
comment	Commentaire de l'ordre
position	Ticket d'une position. Doit être rempli lorsque la position est modifiée ou fermée pour identifier la position. La règle est qu'il est égal au ticket de l'ordre ayant ouvert la position.
position_by	Ticket d'une position opposée. Utilisé lorsqu'une position est fermée par une position opposée d'ouverture pour le même symbole dans la direction opposée.

Lors de la modification ou la clôture d'une position avec le système de hedging, assurez-vous de spécifier son ticket (MqlTradeRequest::position). Le ticket peut également être spécifié dans le système de netting, bien qu'une position soit identifié avec le nom du symbole.

Lors de l'envoi d'ordres pour effectuer des opérations de trading, il est nécessaire d'utiliser la fonction OrderSend(). Pour chaque opération, il est nécessaire de spécifier des champs obligatoires, des champs optionnels peuvent également être spécifiés. Il y a sept cas possibles d'envoi d'ordre :

Exécution de la Demande

C'est un ordre de trading pour ouvrir une position dans le mode Exécution de la Demande (trade upon requested prices). Il est demandé de spécifier les 9 champs suivants :

- action
- symbol
- volume
- prix
- sl
- tp
- deviation
- type

© 2000-2024, MetaQuotes Ltd.

- type_filling

Il est également possible de spécifier les valeurs des champs "magic" et "comment".

Exécution Instantanée

C'est un ordre de trading pour ouvrir une position dans le mode Exécution Instantanée (trade by current prices). Il est demandé de spécifier les 9 champs suivants :

- action
- symbol
- volume
- prix
- sl
- tp
- deviation
- type
- type_filling

Il est également possible de spécifier les valeurs des champs "magic" et "comment".

Execution au Marché

C'est un ordre de trading pour ouvrir une position dans le mode Exécution au Marché. Il est demandé de spécifier les 5 champs suivants :

- action
- symbol
- volume
- type
- type_filling

Il est également possible de spécifier les valeurs des champs "magic" et "comment".

Exécution en Bourse

C'est un ordre de trading pour ouvrir une position dans le mode Exécution en Bourse. Il est demandé de spécifier les 5 champs suivants :

- action
- symbol
- volume
- type
- type_filling

Il est également possible de spécifier les valeurs des champs "magic" et "comment".

Exemple de l'opération de trading TRADE_ACTION_DEAL pour ouvrir une position Buy :

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber de l'
//+------------------------------------------------------------------+
//| Ouverture d'une position Buy                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trading et le résultat de la demande
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- paramètres de la demande
   request.action   =TRADE_ACTION_DEAL;                     // type de l'opération de
   request.symbol   =Symbol();                              // symbole
   request.volume   =0.1;                                   // volume de 0.1 lot
   request.type     =ORDER_TYPE_BUY;                        // type de l'ordre
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // prix d'ouverture
   request.deviation=5;                                     // déviation du prix auto
   request.magic    =EXPERT_MAGIC;                          // MagicNumber de l'ordre
//--- envoi la demande
   if(!OrderSend(request,result))
      PrintFormat("OrderSend erreur %d",GetLastError());     // en cas d'erreur d'envo
//--- informations de l'opération
   PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,result.deal
  }
//+------------------------------------------------------------------+
```

Exemple de l'opération de trading TRADE_ACTION_DEAL pour ouvrir une position Sell :

```
#define EXPERT_MAGIC 123456   // MagicNumber de l'
//+------------------------------------------------------------------+
//| Ouverture d'une position Sell                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trading et le résultat de la demande
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- paramètres de la demande
   request.action   =TRADE_ACTION_DEAL;                     // type de l'opération de
   request.symbol   =Symbol();                              // symbole
   request.volume   =0.1;                                   // volume de 0.2 lot
   request.type     =ORDER_TYPE_SELL;                        // type de l'ordre
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // prix d'ouverture
   request.deviation=5;                                     // déviation du prix auto
   request.magic    =EXPERT_MAGIC;                          // MagicNumber de l'ordre
//--- envoi la demande
   if(!OrderSend(request,result))
      PrintFormat("OrderSend erreur %d",GetLastError());     // en cas d'erreur d'envo
//--- informations de l'opération
   PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,result.deal
  }
<b88>//+------------------------------------------------------------------+
```

Exemple de l'opération de trading TRADE_ACTION_DEAL pour fermer des positions :

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber de l'expert
//+------------------------------------------------------------------+
//| Fermeture de toutes les positions                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trading et le résultat
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // nombre de positions ouvertes
//--- boucle sur toutes les positions ouvertes
   for(int i=total-1; i>=0; i--)
     {
      //--- paramètres de l'ordre
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- affiche les informations de la position
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- si le MagicNumber correspond
      if(magic==EXPERT_MAGIC)
        {
         //--- remise à zéro de la demande et du résultat
         ZeroMemory(request);
         ZeroMemory(result);
         //--- paramètres de l'opration
         request.action   =TRADE_ACTION_DEAL;        // type de l'opération de tradin
         request.position =position_ticket;          // ticket de la position
         request.symbol   =position_symbol;          // symbole
         request.volume   =volume;                   // volume de la position
         request.deviation=5;                        // déviation du prix autorisée
         request.magic    =EXPERT_MAGIC;             // MagicNumber de la position
         //--- définit le prix et le type de l'ordre suivant le type de la position
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
           }
         //--- affiche les informations de clôture
         PrintFormat("Ferme #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- envoi la demande
         if(!OrderSend(request,result))
            PrintFormat("OrderSend erreur %d",GetLastError());  // en cas d'échec de
         //--- informations sur l'opération
         PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,resu
         //---
        }
```

```
      }
   }
//+------------------------------------------------------------------+
```

Modification des SL & TP

Ordre de trading pour modifier les niveaux des prix des StopLoss et/ou TakeProfit. Il est demandé de spécifier les 4 champs suivants :

- action
- symbol
- sl
- tp
- position

Exemple de l'opération de trading TRADE_ACTION_SLTP pour modifier les valeurs des Stop Loss et Take Profit d'une position ouverte :

```
#define EXPERT_MAGIC 123456  // MagicNumber de l' expert
//+------------------------------------------------------------------+
//| Modification du Stop Loss et du Take Profit d'une position       |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trade et le résultat
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // nombre de positions ouvertes
//--- boucle sur toutes les positions ouvertes
   for(int i=0; i<total; i++)
     {
      //--- paramètres de l'ordre
      ulong  position_ticket=PositionGetTicket(i);// ticket de la position
      string position_symbol=PositionGetString(POSITION_SYMBOL); // symbole
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // nombre d
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // MagicNumber de la position
      double volume=PositionGetDouble(POSITION_VOLUME);    // volume de la position
      double sl=PositionGetDouble(POSITION_SL);  // Stop Loss de la position
      double tp=PositionGetDouble(POSITION_TP);  // Take Profit de la position
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- affiche quelques informations sur la position
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- si le MagicNumber correspond et que le Stop Loss et le Take Profit ne sont
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2024, MetaQuotes Ltd.

```
         //--- calcule les niveaux de prix actuels
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_I
         double price_level;
         //--- si la distance de décalage minimum autorisée en points du prix de clotu
         if(stop_level<=0)
            stop_level=150; // définit la distance de décalage de 150 points du priwx
         else
            stop_level+=50; // définit la distance de décalage à (SYMBOL_TRADE_STOPS_I

         //--- calcul et arrondi des valeurs du Stop Loss et du Take Profit
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(bid+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(ask-price_level,digits);
           }
         //--- remise à zéro de la demande et du résultat
         ZeroMemory(request);
         ZeroMemory(result);
         //--- définition des paramètres de l'opération
         request.action  =TRADE_ACTION_SLTP; // type de l'opération de trading
         request.position=position_ticket;   // ticket de la position
         request.symbol=position_symbol;     // symbole
         request.sl      =sl;                // Stop Loss de la position
         request.tp      =tp;                // Take Profit de la position
         request.magic=EXPERT_MAGIC;         // MagicNumber de la position
         //--- affiche des informations sur la modification
         PrintFormat("Modification de #%I64d %s %s",position_ticket,position_symbol,Er
         //--- envoi de la demande
         if(!OrderSend(request,result))
            PrintFormat("OrderSend erreur %d",GetLastError());  // en cas d'échec de l
         //--- informations sur l'opération
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
      }
   }
//+------------------------------------------------------------------+
```

Ordre en Attente

Ordre pour placer un ordre en attente. Il est nécessaire de spécifier les 11 champs suivants :

- action
- symbol
- volume
- prix
- stoplimit
- sl
- tp
- type

© 2000-2024, MetaQuotes Ltd.

- type_filling
- type_time
- expiration

Il est également possible de spécifier les valeurs des champs "magic" et "comment".

Exemple de l'opération de trading TRADE_ACTION_PENDING pour placer un ordre en attente :

© 2000-2024, MetaQuotes Ltd.

```
#property description "Exemple de placement d'ordres en attente"
#property script_show_inputs
#define EXPERT_MAGIC 123456                              // MagicNumber de l'expert
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;   // type de l'ordre
//+------------------------------------------------------------------+
//| Placement d'ordres en attente                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trade et le résultat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- paramètres pour placer un ordre en attente
   request.action   =TRADE_ACTION_PENDING;                          // type de trac
   request.symbol   =Symbol();                                      // symbole
   request.volume   =0.1;                                           // volume de 0.
   request.deviation=2;                                             // déviation au
   request.magic    =EXPERT_MAGIC;                                  // MagicNumber
   int offset = 50;                                                 // décalage du
   double price;                                                    // prix de décl
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);             // valeur du po
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);             // nombre de dé
   //--- vérification du type de l'opération
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                      // type de l'or
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;    // prix d'ouver
      request.price    =NormalizeDouble(price,digits);             // prix d'ouver
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                      // type de l'o
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;    // prix d'ouve
      request.price    =NormalizeDouble(price,digits);              // prix d'ouve
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                           // type de l'o
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point; // prix d'ouve
      request.price=NormalizeDouble(price,digits);                 // prix d'ouve
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                      // type de l'o
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;    // prix d'ouve
      request.price    =NormalizeDouble(price,digits);             // prix d'ouve
     }
   else Alert("Cet exemple ne permet de ne placer que des ordres en attente");   // si
//--- envoie la demande
   if(!OrderSend(request,result))
      PrintFormat("OrderSend erreur %d",GetLastError());            // en cas d'é
//--- informations sur l'opération
   PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,result.deal
  }
//+------------------------------------------------------------------+
```

Modifier des Ordres en Attente

Ordre pour modifier les prix d'un ordre en attente. Il est nécessaire de spécifier les 7 champs suivants :

© 2000-2024, MetaQuotes Ltd.

- action
- ordre
- prix
- sl
- tp
- type_time
- expiration

Exemple de l'opération de trading TRADE_ACTION_MODIFY pour modifier les niveaux de prix d'ordres en attente :

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber de l'expert
//+------------------------------------------------------------------+
//| Modification des ordres en attente                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trade et le résultat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // nombre total d'ordres en attente placés
//--- boucle sur tous les ordres en attente placés
   for(int i=0; i<total; i++)
     {
      //--- paramètres pour l'ordre
      ulong  order_ticket=OrderGetTicket(i);                          // ticket de
      string order_symbol=Symbol();                                   // symbole
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS);  // nombre de
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                      // MagicNumbe
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);             // volume act
      double sl=OrderGetDouble(ORDER_SL);                             // Stop Loss
      double tp=OrderGetDouble(ORDER_TP);                             // Take Profi
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // type de l
      int offset = 50;                                                // décalage d
      double price;                                                   // prix de dé
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);       // valeur du
      //--- affiche des informations sur l'ordre
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  order_ticket,
                  order_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- si le MagicNumber correspond, et que le Stop Loss et le Take Profit ne son
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                        // type de l'op
         request.order = OrderGetTicket(i);                         // ticket de l
         request.symbol   =Symbol();                                // symbole
         request.deviation=5;                                       // déviation au
        //--- définition du niveau de prix, du Take Profit et du Stop Loss de l'ordre
         if(type==ORDER_TYPE_BUY_LIMIT)
           {
            price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prix d
           }
         else if(type==ORDER_TYPE_SELL_LIMIT)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prix d
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
```

```
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prix c
           }
         else if(type==ORDER_TYPE_SELL_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prix c
           }
         //--- envoi de la demande
         if(!OrderSend(request,result))
            PrintFormat("OrderSend erreur %d",GetLastError());  // en cas d'échec de l
         //--- informations sur l'opération
         PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,resul
         //--- remise à zéro de la demande et du résultat
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

Supprimer un Ordre en Attente

Ordre pour supprimer un ordre en attente. Il est nécessaire de spécifier les 2 champs suivants :

- action
- ordre

Exemple de l'opération de trading TRADE_ACTION_REMOVE pour supprimer des ordres en attente :

```
#define EXPERT_MAGIC 123456  // MagicNumber de l'expert
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Suppression des ordres en attente                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare et initialise la demande de trading et le résultat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // nombre total d'ordre en attente placés
//--- boucle sur tous les ordres en attente placés
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // Ticket de l'ordre
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // MagicNumber de l'orc
      //--- si le MagicNumber correspond
      if(magic==EXPERT_MAGIC)
        {
         //--- remise à zéro de la demande et du résultat
         ZeroMemory(request);
         ZeroMemory(result);
         //--- paramètres de l'opération
         request.action=TRADE_ACTION_REMOVE;                   // type de l'opération
         request.order = order_ticket;                         // Ticket de l'ordre
         //--- envoi la demande
         if(!OrderSend(request,result))
            PrintFormat("OrderSend erreur %d",GetLastError());  // en cas d'échec de l
         //--- informations sur l'opération
         PrintFormat("retcode=%u  transaction=%I64u  ordre=%I64u",result.retcode,resul
        }
     }
  }
//+------------------------------------------------------------------+
```

Voir aussi

Structures et Classes, fonctions de Trading, Propriétés de l'Ordre

© 2000-2024, MetaQuotes Ltd.

La Structure des Résultats de la Vérification d'une Demande de Trade (MqlTradeCheckResult)

Avant d'envoyer une demande pour une opération de trading à un serveur de trades, il est recommandé de la vérifier. La vérification est effectuée avec la fonction OrderCheck(), à laquelle la demande vérifiée et une variable de type MqlTradeCheckResult est passée. Le résultat sera écrit dans cette variable.

```
struct MqlTradeCheckResult
  {
   uint         retcode;             // Code de retour
   double       balance;             // Solde après l'exécution de la transaction
   double       equity;              // Fonds après l'exécution de la transaction
   double       profit;              // Profit flottant
   double       margin;              // Exigences de marge
   double       margin_free;         // Marge libre
   double       margin_level;        // Niveau de marge
   string       comment;             // Commentaire sur le code de retour (descriptio
  };
```

Description des Champs

Champ	Description
retcode	Code de retour
balance	Valeur de la balance après l'éxécution du trade
equity	Valeur des fonds qui sera le résultat de l'exécution de l'opération de trading
profit	Valeur du profit flottant après l'éxécution du trade
margin	Marge requise pour le trade
margin_free	Marge libre après l'éxécution du trade
margin_level	Niveau de marge fixé après l'éxécution du trade
comment	Commentaire du code de retour, description de l'erreur

Voir aussi

Structure de Demande de Trade, Structure pour les Prix Actuels, OrderSend, OrderCheck

© 2000-2024, MetaQuotes Ltd.

La Structure de Résultat d'une Demande de Trade (MqlTradeResult)

Comme résultat d'une demande de trade, un serveur de trades retourne des informations sur le résultat du traitement de la demande de trade sous la forme d'une structure spéciale prédéfinie de type MqlTradeResult.

```
struct MqlTradeResult
  {
   uint     retcode;          // Code de retour de l'opération
   ulong    deal;             // Ticket de la transaction si elle est effectuée
   ulong    order;            // Ticket de l'ordre, s'il est placé
   double   volume;           // Volume de la transaction confirmé par le courtier
   double   price;            // Prix de la transaction confirmé par le courtier
   double   bid;              // Prix Bid actuel
   double   ask;              // Prix Ask actuel
   string   comment;          // Commentaire du courtier pour l'opération (par défaut,
   uint     request_id;       // Identifiant de la demande renvoyé par le terminal per
   int      retcode_external; // Code de retour d'un système de trading externe
  };
```

Description des champs

Champ	Description
retcode	Code de retour d'un serveur de trades
deal	Ticket de la transaction si la transaction a été effectuée. Il est disponible pour une opération de trading de type TRADE_ACTION_DEAL
ordre	Ticket de l'ordre, si un ticket a été placé. Il est disponible pour une opération de type TRADE_ACTION_PENDING
volume	Volume de la transaction, confirmé par le courtier. Il dépend du type de remplissage de l'ordre
prix	Prix de la transaction confirmé par le courtier. Il dépend de la *déviation* de la demande de trade et/ou de l'opération de trading
bid	Le prix du marché Bid actuel (requote price)
ask	Le prix du marché Ask actuel (requote price)
comment	Le commentaire du courtier pour l'opération (par défaut, il est rempli avec la description du code de retour du serveur de trades)
request_id	Identifiant de la demande défini par le terminal lors de l'envoi au serveur de trades
retcode_external	Le code de l'erreur retourné par un système de trading externe. L'utilisation et les types de ces erreurs dépendent du courtier et du système de trading externe auquel les opérations de trading sont envoyées.

© 2000-2024, MetaQuotes Ltd.

Le résultat de l'opération de trading est retourné dans une variable de type MqlTradeResult, qui est passée en second argument à OrderSend() pour effectuer des opérations de trading.

Le terminal détermine l'identifiant de la demande dans le champ request_id lors de son envoi au serveur de trades en utilisant les fonctions OrdersSend() et OrderSendAsync(). Le terminal reçoit les messages à propos des transactions effectuées depuis le serveur de trades et les soumet pour le traitement via la fonction OnTradeTransaction() contenant les paramètres suivantes :

- la description de la transaction de trading dans une structure MqlTradeTransaction ;
- la description de la demande de trade envoyée par la fonction OrderSend() ou par la fonction OrdersSendAsync(). L'identifiant de la demande est envoyée par le terminal au serveur de trades, tandis que la demande elle-même et son identifiant request_id sont stockés dans la mémoire du terminal ;
- le résultat de l'exécution de la demande de trade dans une structure MqlTradeResult avec le champ request_id contenant l'identifiant de cette demande.

La fonction OnTradeTransaction() reçoit 3 paramètres en entrée mais les deux derniers ne sont utilisés que pour les transactions de type TRADE_TRANSACTION_REQUEST. Dans tous les autres cas, les données de la demande de trade et le résultat de son exécution ne sont pas remplis. Un exemple d'analyse des paramètres est disponible dans le chapitre Structure d'une Demande de Trade.

L'ajout du request_id par le terminal pour la demande de trade lors de son envoi au serveur a été principalement introduit pour l'utilisation de la fonction asynchrone OrderSendAsync(). Cet identifiant permet d'associer l'action effectuée (appels aux fonctions OrderSend ou OrderSendAsync) avec le résultat de cette action envoyé à la foncion OnTradeTransaction().

Exemple :

```
//+------------------------------------------------------------------------------------
//| Envoi d'une demande de trading avec le traitement du résultat
//+------------------------------------------------------------------------------------
bool MyOrderSend(MqlTradeRequest request,MqlTradeResult result)
  {
//--- réinitialise le dernier code d'erreur à zéro
   ResetLastError();
//--- envoie la demande
   bool success=OrderSend(request,result);
//--- si le résultat échoue, essaye de savoir pourquoi
   if(!success)
     {
      int answer=result.retcode;
      Print("TradeLog : Echec de la demande de trade. Erreur = ",GetLastError());
      switch(answer)
        {
         //--- requote
         case 10004:
           {
            Print("TRADE_RETCODE_REQUOTE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
```

© 2000-2024, MetaQuotes Ltd.

```
        }
      //--- l'ordre n'est pas accepté par le serveur
      case 10006:
        {
         Print("TRADE_RETCODE_REJECT");
         Print("request.price = ",request.price,"   result.ask = ",
               result.ask," result.bid = ",result.bid);
         break;
        }
      //--- prix invalide
      case 10015:
        {
         Print("TRADE_RETCODE_INVALID_PRICE");
         Print("request.price = ",request.price,"   result.ask = ",
               result.ask," result.bid = ",result.bid);
         break;
        }
      //--- SL et/ou TP invalides
      case 10016:
        {
         Print("TRADE_RETCODE_INVALID_STOPS");
         Print("request.sl = ",request.sl," request.tp = ",request.tp);
         Print("result.ask = ",result.ask," result.bid = ",result.bid);
         break;
        }
      //--- volume invalide
      case 10014:
        {
         Print("TRADE_RETCODE_INVALID_VOLUME");
         Print("request.volume = ",request.volume,"   result.volume = ",
               result.volume);
         break;
        }
      //--- pas assez d'argent pour l'opération de trading
      case 10019:
        {
         Print("TRADE_RETCODE_NO_MONEY");
         Print("request.volume = ",request.volume,"   result.volume = ",
               result.volume,"   result.comment = ",result.comment);
         break;
        }
      //--- une autre raison, affiche le code de retour du serveur
      default:
        {
         Print("Autre réponse = ",answer);
        }
    }
  //--- notification de l'échec de la demande de trading en retournant false
  return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- OrderSend() retourne true - répéte la même réponse
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

Structure d'une Transaction de Trading (MqlTradeTransaction)

Lors de l'exécution de certains actions définies sur un compte de trading, son état change. De telles actions incluent :

- L'envoi d'une demande de trading depuis n'importe application MQL5 dans le terminal client avec les fonctions OrderSend et OrderSendAsync et son exécution ;
- L'envoi d'une demande de trading vie l'interface graphique du terminal et son exécution ;
- L'activation des ordres en attente et des stops sur le serveur ;
- Effectuer des opérations du côté du serveur de trades.

Les transactions de trading suivantes sont effectuées suite à ces actions :

- traitement d'une demande de trading ;
- changement des ordres ouverts ;
- changement de l'historique des ordres ;
- changement de l'historique des transactions ;
- changement des positions.

Par exemple, lors de l'envoi d'un ordre d'achat, il est pris en compte par le gestionnaire, un ordre d'achat correspondant est créé pour le compte, l'ordre est ensuite exécuté et supprimé de la liste des ordres ouverts, puis il est ajouté à l'historique des ordres, une transaction correspondante est ajoutée à l'hisorique et une nouvelle position est créée. Toutes ces actions sont des transactions de trading.

Le gestionnaire spécial OnTradeTransaction() est fourni dans MQL5 pour récupérer les transactions de trading appliquées à un compte. Le premier paramètre du gestionnaire est la structure MqlTradeTransaction décrivant les transactions de trading.

```
struct MqlTradeTransaction
  {
   ulong                        deal;             // Ticket de la transaction
   ulong                        order;            // Ticket de l'ordre
   string                       symbol;           // Symbole de trading name
   ENUM_TRADE_TRANSACTION_TYPE  type;             // Type de la transaction de tradir
   ENUM_ORDER_TYPE              order_type;       // Type d'ordre
   ENUM_ORDER_STATE             order_state;      // Etat de l'ordre
   ENUM_DEAL_TYPE               deal_type;        // Type de la transaction
   ENUM_ORDER_TYPE_TIME         time_type;        // Durée de vie de l'ordre
   datetime                     time_expiration;  // Heure d'expiration de l'ordre
   double                       price;            // Prix
   double                       price_trigger;    // Prix d'activation de l'ordre Sto
   double                       price_sl;         // Niveau du Stop Loss
   double                       price_tp;         // Niveau du Take Profit
   double                       volume;           // Volume en lots
   ulong                        position;         // Ticket de la position
   ulong                        position_by;      // Ticket d'une position opposée
  };
```

© 2000-2024, MetaQuotes Ltd.

Description des Champs

Champ	Description
deal	Ticket de la transaction.
ordre	Ticket de l'ordre.
symbol	Le nom du symbole de trading, pour lequel une transaction est effectuée.
type	Type de la transaction de trading. La valeur peut être l'une des valeur de l'énumération ENUM_TRADE_TRANSACTION_TYPE.
order_type	Type de l'ordre de trading. La valeur peut être l'une des valeur de l'énumération ENUM_ORDER_TYPE.
order_state	Etat de l'ordre de trading. La valeur peut être l'une des valeur de l'énumération ENUM_ORDER_STATE.
deal_type	Type de la transaction. La valeur peut être l'une des valeur de l'énumération ENUM_DEAL_TYPE.
time_type	Durée de vie de l'ordre. La valeur peut être l'une des valeur de l'énumération ENUM_ORDER_TYPE_TIME.
time_expiration	Expiration de l'ordre en attente (pour les ordres de type ORDER_TIME_SPECIFIED et ORDER_TIME_SPECIFIED_DAY).
prix	Prix. Suivant le type de la transaction de trading, il peut être le prix d'un ordre, d'une transaction ou d'une position.
price_trigger	Prix stop (d'activation) d'un ordre stop limit (ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT).
price_sl	Prix du Stop Loss. Suivant le type de la transaction de trading, il peut être relatif à un ordre, à une transaction ou à une position.
price_tp	Prix du Take Profit. Suivant le type de la transaction de trading, il peut être relatif à un ordre, à une transaction ou à une position.
volume	Volume en lots. Suivant le type de la transaction de trading, il peut indiquer le volume actuel d'un ordre, d'une transaction ou d'une position.
position	Le ticket de la position affectée par la transaction.
position_by	Le ticket de la position opposée. Utilisé lors de la fermeture d'une position par une position opposée, c'est à dire par une position sur le même symbole ouvert dans une direction opposée.

Le principal paramètre pour l'analyse de la transaction reçue est son type spécifié dans le champ **type**. Par exemple, si une transaction est de type TRADE_TRANSACTION_REQUEST (un résultat du traitement de la demande a été reçue du serveur), la structure n'a qu'un seul champ rempli - **type**. Les autres champs ne sont pas analysés. Dans ce cas, nous pouvons analyser deux paramètres supplémentaires **request** et **result** soumis par le gestionnaire OnTradeTransaction(), comme montré ci-dessous.

© 2000-2024, MetaQuotes Ltd.

Avec les données sur le type de l'opération de trading, vous pouvez décider de l'analyse de l'état actuel des ordres, des positions et des transactions sur un compte de trading. N'oubliez pas qu'une demande de trade envoyée au serveur depuis le terminal peut générer plusieurs nouvelles transactions. La priorité de leur arrivée au terminal n'est pas garantie.

La structure MqlTradeTransaction est remplie de différentes façons suivant le type (ENUM_TRADE_TRANSACTION_TYPE) de la transaction :

TRADE_TRANSACTION_ORDER_* et TRADE_TRANSACTION_HISTORY_*

Les champs suivants de la structure MqlTradeTransaction sont remplis pour les transactions de trading relatives au traitement des ordres d'ouverture (TRADE_TRANSACTION_ORDER_ADD, TRADE_TRANSACTION_ORDER_UPDATE and TRADE_TRANSACTION_ORDER_DELETE) et à l'historique des ordres (TRADE_TRANSACTION_HISTORY_ADD, TRADE_TRANSACTION_HISTORY_UPDATE, TRADE_TRANSACTION_HISTORY_DELETE) :

- order - ticket de l'ordre ;
- symbol - nom du symbole de l'ordre ;
- type - type de la transaction ;
- order_type - type de l'ordre ;
- orders_state - état actuel de l'ordre ;
- time_type - type d'expiration de l'ordre ;
- time_expiration - date/heure d'expiration de l'ordre (pour les ordres ayant les types d'expiration ORDER_TIME_SPECIFIED et ORDER_TIME_SPECIFIED_DAY) ;
- price - prix de l'ordre spécifié par un client ;
- price_trigger - prix stop de l'ordre stop limit (seulement pour ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT) ;
- price_sl - prix du Stop Loss (rempli, si spécifié dans l'ordre) ;
- price_tp - prix du Take Profit (rempli, si spécifié dans l'ordre) ;
- volume - volume actuel de l'ordre (non rempli). Le volume initial de l'ordre peut être trouvé dans l'historique des ordres avec la fonction HistoryOrders* .
- position - le ticket de la position qui a été ouverte, modifiée ou fermée comme étant le résultat de l'exécution de l'ordre. Il n'est rempli que pour les ordres au marché, il n'est pas rempli pour TRADE_TRANSACTION_ORDER_ADD.
- position_by - le ticket de la position opposée. Il n'est rempli que pour les ordres 'close by' (pour fermer une position par une position opposée).

TRADE_TRANSACTION_DEAL_*

Les champs suivants de la structure MqlTradeTransaction sont remplis pour les transactions relatives au traitement des deals (TRADE_TRANSACTION_DEAL_ADD, TRADE_TRANSACTION_DEAL_UPDATE et TRADE_TRANSACTION_DEAL_DELETE) :

- deal - ticket de la transaction ;
- order - ticket de l'ordre, sur la base duquel une transaction a été effectuée ;
- symbol - nom du symbole de la transaction ;
- type - type de la transaction ;
- deal_type - type de la transaction;
- price - prix de la transaction ;
- price_sl - prix du Stop Loss (rempli, si spécifié dans l'ordre, sur la base duquel une transaction a été effectuée) ;

© 2000-2024, MetaQuotes Ltd.

- price_tp - prix du Take Profit (rempli, si spécifié dans l'ordre, sur la base duquel une transaction a été effectuée) ;
- volume - volume de la transaction en lots.
- position - le ticket de la position qui a été ouverte, modifiée ou fermée comme étant le résultat de l'exécution du deal.
- position_by - le ticket de la position opposée. Il n'est rempli que pour les deals 'out by' (fermeture d'une position par une position opposée).

TRADE_TRANSACTION_POSITION

Les champs suivants de la structure MqlTradeTransaction sont remplis pour les transactions de trading relatives au changement des positions non connectées à l'exécution des transactions (TRADE_TRANSACTION_POSITION):

- symbol - nom du symbole de la position ;
- type - type de la transaction ;
- deal_type - type de la position (DEAL_TYPE_BUY ou DEAL_TYPE_SELL);
- price - prix d'ouverture moyen pondéré de la position ;
- price_sl - prix du Stop Loss ;
- price_tp - prix du Take Profit ;
- volume - volume de la position en lots, s'il a été changéif it has been changed.

Le changement de la position (ajout, changement ou clôture), comme étant le résultat de l'exécution d'une transaction, ne mène pas à une transaction TRADE_TRANSACTION_POSITION.

TRADE_TRANSACTION_REQUEST

Seul un champ de la structure MqlTradeTransaction est rempli pour les transactions décrivant le fait qu'une demande de trade a été traitée par un serveur et que le résultat a été reçu (TRADE_TRANSACTION_REQUEST):

- type - type de la transaction ;

Un seul champ (type de la transaction de trading) doit être analysé pour ces transactions. Les deuxième et troisième paramètres de la fonction OnTradeTransaction (demande et résultat) doivent être analysés pour obtenir des données supplémentaires.

Exemple :

```
input int MagicNumber=1234567;

//--- active la classe de trading CTrade et déclare une variable de cette classe
#include <Trade\Trade.mqh>
CTrade trade;
//--- flags pour installer et supprimer l'ordre en attente
bool pending_done=false;
bool pending_deleted=false;
//--- le ticket de l'ordre en attente sera stocké ici
ulong order_ticket;
//+------------------------------------------------------------------------------------
//| Fonction d'initialisation de l'expert
//+------------------------------------------------------------------------------------
int OnInit()
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- définit le MagicNumber pour marquer nos ordres
   trade.SetExpertMagicNumber(MagicNumber);
//--- les demandes de trade seront envoyées en mode asynchrone en utilisant la fonctio
   trade.SetAsyncMode(true);
//--- initialise la variable à zéro
   order_ticket=0;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------
//| Fonction de tick de l'Expert
//+------------------------------------------------------------------------------------
void OnTick()
  {
//--- installation d'un ordre en attente
   if(!pending_done)
     {
      double ask=SymbolInfoDouble(_Symbol,SYMBOL_ASK);
      double buy_stop_price=NormalizeDouble(ask+1000*_Point,(int)SymbolInfoInteger(_Sy
      bool res=trade.BuyStop(0.1,buy_stop_price,_Symbol);
      //--- si la fonction BuyStop() a été effectuée avec succès
      if(res)
        {
         pending_done=true;
         //--- récupère le résultat de l'envoi de la demande depuis ctrade
         MqlTradeResult trade_result;
         trade.Result(trade_result);
         //--- récupère le request_id pour la demande envoyée
         uint request_id=trade_result.request_id;
         Print("La demande pour un ordre en attente a été envoyée. Request_ID=",reques
         //--- stockage du ticket de l'ordre (sera zéro si le mode asynchrone est util
         order_ticket=trade_result.order;
         //--- tout est fait, sortie rapide du handler OnTick()
         return;
        }
     }
//--- supprime l'ordre en attente
   if(!pending_deleted)
      //--- vérification supplémentaire
      if(pending_done && (order_ticket!=0))
        {
         //--- essaye de supprimer l'ordre en attente
         bool res=trade.OrderDelete(order_ticket);
         Print("OrderDelete=",res);
         //--- lorsque la demande de suppression est envoyée avec succès
         if(res)
           {
            pending_deleted=true;
```

```
            //--- récupère le résultat de l'exécution de la demande
            MqlTradeResult trade_result;
            trade.Result(trade_result);
            //--- récupère le request ID du résultat
            uint request_id=trade_result.request_id;
            //--- affichage dans le Journal
            Print("La demande de suppression de l'ordre en attente #",order_ticket,
                  " a été envoyée. Request_ID=",request_id,
                  "\r\n");
            //--- récupère le ticket de l'ordre depuis le résultat de la demande
            order_ticket=trade_result.order;
           }
        }
//---
  }
//+------------------------------------------------------------------------------
//| Fonction TradeTransaction
//+------------------------------------------------------------------------------
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- récupère le type de la transaction comme une valeur de l'énumération
   ENUM_TRADE_TRANSACTION_TYPE type=(ENUM_TRADE_TRANSACTION_TYPE)trans.type;
//--- si la transaction est le résultat du traitement de la demande, seul son nom est
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      Print(EnumToString(type));
      //--- affiche le nom de la demande gérée
      Print("------------RequestDescription\r\n",RequestDescription(request));
      //--- affiche la description du résultat de la demande
      Print("------------ResultDescription\r\n",TradeResultDescription(result));
      //--- stocke le ticket de l'ordre pour sa suppression au prochain appel de OnTic
      if(result.order!=0)
        {
         //--- supprime cet ordre par son ticket au prochain appel de OnTick()
         order_ticket=result.order;
         Print(" Ticket de l'ordre en attente ",order_ticket,"\r\n");
        }
     }
   else // affiche la description complète pour les transactions d'un autre type
//--- affiche la description de la transaction reçue dans le Journal
      Print("------------TransactionDescription\r\n",TransactionDescription(trans));

//---
  }
//+------------------------------------------------------------------------------
//| Retourne la description textuelle de la transaction
//+------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
string TransactionDescription(const MqlTradeTransaction &trans)
  {
//---
   string desc=EnumToString(trans.type)+"\r\n";
   desc+="Symbole : "+trans.symbol+"\r\n";
   desc+="Ticket de la transaction : "+(string)trans.deal+"\r\n";
   desc+="Type de la transaction : "+EnumToString(trans.deal_type)+"\r\n";
   desc+="Ticket de l'ordre : "+(string)trans.order+"\r\n";
   desc+="Type de l'ordre : "+EnumToString(trans.order_type)+"\r\n";
   desc+="Etat de l'ordre : "+EnumToString(trans.order_state)+"\r\n";
   desc+="Type d'expiration de l'ordre : "+EnumToString(trans.time_type)+"\r\n";
   desc+="Expiration de l'ordre : "+TimeToString(trans.time_expiration)+"\r\n";
   desc+="Prix : "+StringFormat("%G",trans.price)+"\r\n";
   desc+="Déclenchement du prix : "+StringFormat("%G",trans.price_trigger)+"\r\n";
   desc+="Stop Loss : "+StringFormat("%G",trans.price_sl)+"\r\n";
   desc+="Take Profit : "+StringFormat("%G",trans.price_tp)+"\r\n";
   desc+="Volume : "+StringFormat("%G",trans.volume)+"\r\n";
   desc+="Position : "+(string)trans.position+"\r\n";
   desc+="Position by : "+(string)trans.position_by+"\r\n";
//--- retourne la chaîne de caractères obtenue
   return desc;
  }
//+------------------------------------------------------------------------------------
//| Retourne la description textuelle de la demande de trade
//+------------------------------------------------------------------------------------
string RequestDescription(const MqlTradeRequest &request)
  {
//---
   string desc=EnumToString(request.action)+"\r\n";
   desc+="Symbole : "+request.symbol+"\r\n";
   desc+="Magic Number : "+StringFormat("%d",request.magic)+"\r\n";
   desc+="Ticket de l'ordre : "+(string)request.order+"\r\n";
   desc+="Type de l'ordre : "+EnumToString(request.type)+"\r\n";
   desc+="Remplissage de l'ordre : "+EnumToString(request.type_filling)+"\r\n";
   desc+="Type d'expiration de l'ordre : "+EnumToString(request.type_time)+"\r\n";
   desc+="Expiration de l'ordre : "+TimeToString(request.expiration)+"\r\n";
   desc+="Prix : "+StringFormat("%G",request.price)+"\r\n";
   desc+="Déviation en points : "+StringFormat("%G",request.deviation)+"\r\n";
   desc+="Stop Loss : "+StringFormat("%G",request.sl)+"\r\n";
   desc+="Take Profit : "+StringFormat("%G",request.tp)+"\r\n";
   desc+="Stop Limit : "+StringFormat("%G",request.stoplimit)+"\r\n";
   desc+="Volume : "+StringFormat("%G",request.volume)+"\r\n";
   desc+="Commentaire : "+request.comment+"\r\n";
//--- retourne la chaîne de caractères obtenue
   return desc;
  }
//+------------------------------------------------------------------------------------
//| Retourne la description textuelle du résultat du traitement de la demande
//+------------------------------------------------------------------------------------
```

© 2000-2024, MetaQuotes Ltd.

```
string TradeResultDescription(const MqlTradeResult &result)
  {
//---
   string desc="Retcode "+(string)result.retcode+"\r\n";
   desc+="Request ID : "+StringFormat("%d",result.request_id)+"\r\n";
   desc+="Ticket de l'ordre : "+(string)result.order+"\r\n";
   desc+="Ticket de la transaction : "+(string)result.deal+"\r\n";
   desc+="Volume : "+StringFormat("%G",result.volume)+"\r\n";
   desc+="Prix : "+StringFormat("%G",result.price)+"\r\n";
   desc+="Ask : "+StringFormat("%G",result.ask)+"\r\n";
   desc+="Bid : "+StringFormat("%G",result.bid)+"\r\n";
   desc+="Commentaire : "+result.comment+"\r\n";
//--- retourne la chaîne de caractères obtenue
   return desc;
  }
```

Voir aussi

Types de Transaction de Trading, OnTradeTransaction()

© 2000-2024, MetaQuotes Ltd.

La Structure pour Retourner les Prix Courants (MqlTick)

C'est la structure pour le stockage des derniers prix du symbole. Elle est conçue pour récupérer rapidement les informations les plus courantes des prix.

```
struct MqlTick
  {
   datetime     time;          // Heure de la dernière mise à jour des prix
   double       bid;           // Prix Bid actuel
   double       ask;           // Prix Ask actuel
   double       last;          // Prix de la dernière transaction (Last)
   ulong        volume;        // Volume pour le prix Last actuel
   long         time_msc;      // Heure de la dernière mise à jour du prix en millisec
   uint         flags;         // Flags du tick
   double       volume_real;   // Volume for the current Last price with greater accu
  };
```

La variable de type MqlTick permet d'obtenir les valeurs Ask, Bid, Last et Volume avec un seul appel à la fonction SymbolInfoTick().

Les paramètres de chaque tick sont remplis indépendemment du fait de changements ou pas par rapport au dernier tick. Il est donc possible d'avoir un prix correct à tout moment dans le passé sans avoir besoin de rechercher les valeurs précédentes dans l'historique des ticks. Par exemple, même si seul le prix Bid change à l'arrivée d'un tick, la structure contient toujours les autres paramètres, incluant le dernier prix Ask, le dernier volume, etc.

Vous pouvez analyser les flags du tick pour savoir exactement quelles données ont été changées :

- TICK_FLAG_BID - le tick a changé le prix Bid
- TICK_FLAG_ASK - le tick a changé le prix Ask
- TICK_FLAG_LAST - le tick a changé le prix Last
- TICK_FLAG_VOLUME - le tick a changé le volume
- TICK_FLAG_BUY - le tick est le résultat d'une transaction d'achat
- TICK_FLAG_SELL - le tick est le résultat d'une transaction de vente

Exemple :

```
void OnTick()
  {
   MqlTick last_tick;
//---
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      Print(last_tick.time,": Bid = ",last_tick.bid,
            " Ask = ",last_tick.ask,"  Volume = ",last_tick.volume);
     }
   else Print("Echec de SymbolInfoTick(), erreur = ",GetLastError());
//---
  }
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Structures et Classes, CopyTicks(), SymbolInfoTick()

© 2000-2024, MetaQuotes Ltd.

Structures du Calendrier Economique

Cette section décrit les structures à utiliser pour travailler avec le calendrier économique disponible directement dans la plateforme MetaTrader. Le calendrier économique est une encyclopédie toute faite présentant des indicateurs macroéconomiques, leurs dates de publication et leur importance. Les valeurs pertinentes des indicateurs macroéconomiques sont envoyées à la plateforme MetaTrader dès le moment de la publication et sont affichées sur un graphique sous forme de balises vous permettant de suivre de manière visuelle les indicateurs requis par pays, par devise et par importance.

Les fonctions du calendrier économique permettent de réaliser l'analyse automatique des événements entrants en fonction de critères d'importance personnalisés en fonction des paires pays/devises nécessaires.

Les descriptions des pays sont effectuées avec la structure MqlCalendarCountry. Elle est utilisée dans les fonctions CalendarCountryById() et CalendarCountries()

```
struct MqlCalendarCountry
  {
   ulong                                id;                    // identifiant du pays
   string                               name;                  // nom textuel du pays
   string                               code;                  // nom du code du pays
   string                               currency;              // code de la devise du
   string                               currency_symbol;       // symbole de la devise
   string                               url_name;              // nom du pays utilisé c
  };
```

Les descriptions des évènements sont effectuées avec la structure MqlCalendarEvent. Elle est utilisée dans les fonctions CalendarEventById(), CalendarEventByCountry() et CalendarEventByCurrency()

```
struct MqlCalendarEvent
  {
   ulong                                id;                    // identifiant de l'évèn
   ENUM_CALENDAR_EVENT_TYPE             type;                  // type de l'évènement,
   ENUM_CALENDAR_EVENT_SECTOR           sector;                // secteur lié à l'évène
   ENUM_CALENDAR_EVENT_FREQUENCY        frequency;             // fréquence de l'évèner
   ENUM_CALENDAR_EVENT_TIMEMODE         time_mode;             // mode de l'heure de l'
   ulong                                country_id;            // identifiant du pays
   ENUM_CALENDAR_EVENT_UNIT             unit;                  // unité de mesure de la
   ENUM_CALENDAR_EVENT_IMPORTANCE       importance;            // importance de l'évène
   ENUM_CALENDAR_EVENT_MULTIPLIER       multiplier;            // multiplicateur de la
   uint                                 digits;                // nombre de décimales
   string                               source_url;            // URL de la source publ
   string                               event_code;            // code de l'évènement
   string                               name;                  // nom du texte de l'éve
  };
```

© 2000-2024, MetaQuotes Ltd.

Les valeurs de l'évènement sont définies avec la structure MqlCalendarValue. Elle est utilisée dans les fonctions CalendarValueById(), CalendarValueHistoryByEvent(), CalendarValueHistory(), CalendarValueLastByEvent() et CalendarValueLast()

```
struct MqlCalendarValue
  {
   ulong                              id;                    // identifiant
   ulong                              event_id;              // identifiant de l'évèr
   datetime                           time;                  // date et heure de l'é
   datetime                           period;                // période de publicatic
   int                                revision;              // révision de l'indicat
   long                               actual_value;          // valeur actuelle en pp
   long                               prev_value;            // valeur précédente en
   long                               revised_prev_value;    // valeur précédente rév
   long                               forecast_value;        // valeur prévisionnelle
   ENUM_CALENDAR_EVENT_IMPACT         impact_type;           // impact potentiel sur
  //--- fonctions pour vérifier les valeurs
   bool                        HasActualValue(void) const;   // renvoie true si actua
   bool                        HasPreviousValue(void) const; // renvoie true si prev_
   bool                        HasRevisedValue(void) const;  // renvoie true si revis
   bool                        HasForecastValue(void) const; // renvoie true si forec
  //--- fonctions pour récupérer les valeurs
   double                      GetActualValue(void) const;   // renvoie actual_value
   double                      GetPreviousValue(void) const; // renvoie prev_value ou
   double                      GetRevisedValue(void) const;  // renvoie revised_prev_
   double                      GetForecastValue(void) const; // renvoie forecast_valu
  };
```

La structure MqlCalendarValue fournit des méthodes pour vérifier et définir les valeurs des champs *actual_value*, *forecast_value*, *prev_value* et *revised_prev_value*. Si aucune valeur n'est spécifiée, le champ stocke LONG_MIN (-9223372036854775808).

Veuillez noter que les valeurs stockées dans ces champs sont multipliées par un million. Cela signifie que lorsque vous recevez des valeurs dans MqlCalendarValue à l'aide des fonctions CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent et CalendarValueLast, vous devez vérifier si les valeurs des champs sont égales à LONG_MIN ; si une valeur est spécifiée dans un champ, vous devez diviser la valeur par 1 000 000 afin d'obtenir la valeur. Une autre méthode pour obtenir les valeurs consiste à vérifier et à obtenir des valeurs en utilisant les fonctions de la structure MqlCalendarValue.

Un exemple de gestion des événements du calendrier :

```
//--- Crée une structure pour stocker des événements de calendrier avec des valeurs ré
struct AdjustedCalendarValue
  {
   ulong                              id;                    // identifiant
   ulong                              event_id;              // identifiant de l'évèr
   datetime                           time;                  // date et heure de l'é
   datetime                           period;                // période de publicatic
   int                                revision;              // révision de l'indicat
```

© 2000-2024, MetaQuotes Ltd.

```
   double                              actual_value;          // valeur actuelle
   double                              prev_value;            // valeur précédente
   double                              revised_prev_value;    // valeur précédente rév
   double                              forecast_value;        // valeur prévisionnelle
   ENUM_CALENDAR_EVENT_IMPACT          impact_type;           // impact potentiel sur
  };
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme de script                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
//--- code pays pour l'UE (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- récupère toutes les valeurs d'événement de l'UE
   MqlCalendarValue values[];
//--- définit les limites de l'intervalle à partir duquel les événements sont pris
   datetime date_from=D'01.01.2021';  // prends tous les évènements de 2021
   datetime date_to=0 ; // 0 signifie tous les événements connus, y compris ceux qui r
//--- demande l'historique des événements de l'UE depuis 2021
   if(!CalendarValueHistory(values, date_from, date_to, EU_code))
     {
      PrintFormat("Erreur ! Échec de l'obtention des événements pour country_code=%s",
      PrintFormat("Code d'erreur : %d", GetLastError());
      return;
     }
   else
      PrintFormat("Valeurs d'événement reçues pour country_code=%s : %d",
                  EU_code, ArraySize(values));
//--- réduit la taille du tableau pour la sortie vers le Journal
   if(ArraySize(values)>5)
      ArrayResize(values, 5);
//--- affiche les valeurs d'événement dans le Journal telles qu'elles sont, sans vérif
   Imprimer("Sortie des valeurs du calendrier telles qu'elles sont");
   ArrayPrint(values);

//--- vérifie les valeurs des champs et les convertit en valeurs réelles
//--- option 1 pour vérifier et obtenir les valeurs
   AdjustedCalendarValue values_adjusted_1[];
   int total=ArraySize(values);
   ArrayResize(values_adjusted_1, total);
//--- copie les valeurs avec vérifications et ajustements
   for(int i=0; i<total; i++)
     {
      values_adjusted_1[i].id=values[i].id;
      values_adjusted_1[i].event_id=values[i].event_id;
      values_adjusted_1[i].time=values[i].time;
      values_adjusted_1[i].period=values[i].period;
      values_adjusted_1[i].revision=values[i].revision;
```

© 2000-2024, MetaQuotes Ltd.

```
      values_adjusted_1[i].impact_type=values[i].impact_type;
      //--- vérifie les valeurs et les divise par 1 000 000
      if(values[i].actual_value==LONG_MIN)
         values_adjusted_1[i].actual_value=double("nan");
      else
         values_adjusted_1[i].actual_value=values[i].actual_value/1000000.;

      if(values[i].prev_value==LONG_MIN)
         values_adjusted_1[i].prev_value=double("nan");
      else
         values_adjusted_1[i].prev_value=values[i].prev_value/1000000.;

      if(values[i].revised_prev_value==LONG_MIN)
         values_adjusted_1[i].revised_prev_value=double("nan");
      else
         values_adjusted_1[i].revised_prev_value=values[i].revised_prev_value/1000000.

      if(values[i].forecast_value==LONG_MIN)
         values_adjusted_1[i].forecast_value=double("nan");
      else
         values_adjusted_1[i].forecast_value=values[i].forecast_value/1000000.;
     }
   Print("La première méthode pour vérifier et obtenir les valeurs du calendrier");
   ArrayPrint(values_adjusted_1);

//--- option 2 pour vérifier et obtenir les valeurs
   AdjustedCalendarValue values_adjusted_2[];
   ArrayResize(values_adjusted_2, total);
//--- copie les valeurs avec vérifications et ajustements
   for(int i=0; i<total; i++)
     {
      values_adjusted_2[i].id=values[i].id;
      values_adjusted_2[i].event_id=values[i].event_id;
      values_adjusted_2[i].time=values[i].time;
      values_adjusted_2[i].period=values[i].period;
      values_adjusted_2[i].revision=values[i].revision;
      values_adjusted_2[i].impact_type=values[i].impact_type;
      //--- vérifie et obtient les valeurs
      if(values[i].HasActualValue())
         values_adjusted_2[i].actual_value=values[i].GetActualValue();
      else
         values_adjusted_2[i].actual_value=double("nan");

      if(values[i].HasPreviousValue())
         values_adjusted_2[i].prev_value=values[i].GetPreviousValue();
      else
         values_adjusted_2[i].prev_value=double("nan");

      if(values[i].HasRevisedValue())
```

© 2000-2024, MetaQuotes Ltd.

```
         values_adjusted_2[i].revised_prev_value=values[i].GetRevisedValue();
      else
         values_adjusted_2[i].revised_prev_value=double("nan");

      if(values[i].HasForecastValue())
         values_adjusted_2[i].forecast_value=values[i].GetForecastValue();
      else
         values_adjusted_2[i].forecast_value=double("nan");
     }
   Print("La deuxième méthode pour vérifier et obtenir les valeurs du calendrier");
   ArrayPrint(values_adjusted_2);

//--- option 3 pour obtenir les valeurs - sans vérifications
   AdjustedCalendarValue values_adjusted_3[];
   ArrayResize(values_adjusted_3, total);
//--- copie les valeurs avec vérifications et ajustements
   for(int i=0; i<total; i++)
     {
      values_adjusted_3[i].id=values[i].id;
      values_adjusted_3[i].event_id=values[i].event_id;
      values_adjusted_3[i].time=values[i].time;
      values_adjusted_3[i].period=values[i].period;
      values_adjusted_3[i].revision=values[i].revision;
      values_adjusted_3[i].impact_type=values[i].impact_type;
      //--- obtient les valeurs sans vérification
      values_adjusted_3[i].actual_value=values[i].GetActualValue();
      values_adjusted_3[i].prev_value=values[i].GetPreviousValue();
      values_adjusted_3[i].revised_prev_value=values[i].GetRevisedValue();
      values_adjusted_3[i].forecast_value=values[i].GetForecastValue();
     }
   Print("La troisième méthode pour obtenir les valeurs du calendrier - sans vérificat
   ArrayPrint(values_adjusted_3);
  }
/*
   Nous avons reçu des valeurs d'événement pour country_code=EU : 1051
  Sortie des valeurs du calendrier telles qu'elles sont
         [id] [event_id]              [time]            [period] [revision]       [act
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0 -922337203
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0
  La première méthode pour vérifier et obtenir les valeurs du calendrier
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
```

© 2000-2024, MetaQuotes Ltd.

```
  La deuxième méthode pour vérifier et obtenir les valeurs du calendrier
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
  La troisième méthode pour obtenir des valeurs de calendrier - sans contrôles
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
*/
```

La fréquence de l'évènement est spécifiée dans la structure MqlCalendarEvent. Les valeurs possibles sont listées **ENUM_CALENDAR_EVENT_FREQUENCY**

ID	Description
CALENDAR_FREQUENCY_NONE	La fréquence de publication n'est pas définie
CALENDAR_FREQUENCY_WEEK	Publié une fois par semaine
CALENDAR_FREQUENCY_MONTH	Publié une fois par mois
CALENDAR_FREQUENCY_QUARTER	Publié une fois par trimestre
CALENDAR_FREQUENCY_YEAR	Publié une fois par an
CALENDAR_FREQUENCY_DAY	Publié une fois par jour

Le type de l'évènement est spécifié dans la structure MqlCalendarEvent. Les valeurs possibles sont listées **ENUM_CALENDAR_EVENT_TYPE**

ID	Description
CALENDAR_TYPE_EVENT	Evènement (réunion, discours, etc.)
CALENDAR_TYPE_INDICATOR	Indicateur
CALENDAR_TYPE_HOLIDAY	Vacances

Le secteur économique lié à l'évènement est spécifié dans la structure MqlCalendarEvent. Les valeurs possibles sont listées **ENUM_CALENDAR_EVENT_SECTOR**

ID	Description
CALENDAR_SECTOR_NONE	Le secteur n'est pas défini

© 2000-2024, MetaQuotes Ltd.

ID	Description
CALENDAR_SECTOR_MARKET	Marché, bourse
CALENDAR_SECTOR_GDP	Produit Intérieur Brut (PIB)
CALENDAR_SECTOR_JOBS	Marché du travail
CALENDAR_SECTOR_PRICES	Prix
CALENDAR_SECTOR_MONEY	Finances
CALENDAR_SECTOR_TRADE	Affaires
CALENDAR_SECTOR_GOVERNMENT	Gouvernement
CALENDAR_SECTOR_BUSINESS	Affaires
CALENDAR_SECTOR_CONSUMER	Consommation
CALENDAR_SECTOR_HOUSING	Logement
CALENDAR_SECTOR_TAXES	Taxes
CALENDAR_SECTOR_HOLIDAYS	Vacances

L'importance de l'évènement est spécifiée dans la structure MqlCalendarEvent. Les valeurs possibles sont listées **ENUM_CALENDAR_EVENT_IMPORTANCE**

ID	Description
CALENDAR_IMPORTANCE_NONE	L'importance n'est pas définie
CALENDAR_IMPORTANCE_LOW	Faible importance
CALENDAR_IMPORTANCE_MODERATE	Importance moyenne
CALENDAR_IMPORTANCE_HIGH	Importance forte

Le type de l'unité de mesure utilisée pour afficher les valeurs de l'événement est spécifiée dans la structure MqlCalendarEvent. Les valeurs possibles sont listées **ENUM_CALENDAR_EVENT_UNIT**

ID	Description
CALENDAR_UNIT_NONE	L'unité de mesure n'est pas définie
CALENDAR_UNIT_PERCENT	Pourcentage
CALENDAR_UNIT_CURRENCY	Monnaie nationale
CALENDAR_UNIT_HOUR	Heures
CALENDAR_UNIT_JOB	Emplois
CALENDAR_UNIT_RIG	Derricks

© 2000-2024, MetaQuotes Ltd.

ID	Description
CALENDAR_UNIT_USD	USD
CALENDAR_UNIT_PEOPLE	Personnes
CALENDAR_UNIT_MORTGAGE	Prêts hypothécaires
CALENDAR_UNIT_VOTE	Votes
CALENDAR_UNIT_BARREL	Barils
CALENDAR_UNIT_CUBICFEET	Pieds cubes
CALENDAR_UNIT_POSITION	Positions nettes non commerciales
CALENDAR_UNIT_BUILDING	Immeubles

Dans certains cas, les valeurs des paramètres économiques nécessitent un multiplicateur défini dans la structure MqlCalendarEvent. Les valeurs possibles du multiplicateur sont définies dans la liste **ENUM_CALENDAR_EVENT_MULTIPLIER**

ID	Description
CALENDAR_MULTIPLIER_NONE	Le multiplicateur n'est pas défini
CALENDAR_MULTIPLIER_THOUSANDS	Milliers
CALENDAR_MULTIPLIER_MILLIONS	Millions
CALENDAR_MULTIPLIER_BILLIONS	Milliards
CALENDAR_MULTIPLIER_TRILLIONS	Milliards

L'impact potentiel de l'événement sur un taux de change national est indiqué dans la structure MqlCalendarValue. Les valeurs possibles sont listées **ENUM_CALENDAR_EVENT_IMPACT**

ID	Description
CALENDAR_IMPACT_NA	Impact non défini
CALENDAR_IMPACT_POSITIVE	Impact positif
CALENDAR_IMPACT_NEGATIVE	Impact négatif

L'heure de l'évènement est spécifiée dans la structure MqlCalendarEvent. Les valeurs possibles sont listées **ENUM_CALENDAR_EVENT_TIMEMODE**

ID	Description
CALENDAR_TIMEMODE_DATETIME	La source publie l'heure exacte de l'évènement
CALENDAR_TIMEMODE_DATE	L'évènement dure toute la journée

© 2000-2024, MetaQuotes Ltd.

ID	Description
CALENDAR_TIMEMODE_NOTIME	La source ne publie aucune l'heure pour l'évènement
CALENDAR_TIMEMODE_TENTATIVE	La source publie le jour de l'évènement, mais pas son heure exacte. L'heure est spécifiée lorsque l'évènement se produit.

Voir aussi

Calendrier Economique

© 2000-2024, MetaQuotes Ltd.

Codes d'Erreurs et Warnings

Cette section contient les descriptions suivantes :

- Codes de retour du serveur de trade – analyse des résultats de la demande de trade renvoyés par la fonction OrderSend() ;
- Warnings du compilateur – codes des messages d'avertissements (warning) générés pendant la compilation ;
- Erreurs de compilation – codes des messages d'erreur générés pendant la compilation ;
- Erreurs d'exécution – codes d'erreur durant l'exécution de programmes mql5, qui peuvent être obtenus avec la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

Codes de Retour du Serveur de Trading

Toutes les demandes pour exécuter des opérations de trading sont envoyées sous la forme d'une structure de demande de trade MqlTradeRequest en utilisant la fonction OrderSend(). Le résultat de l'exécution de la fonction est placé dans la structure MqlTradeResult, dont le champ *retcode* contient le code de retour du serveur de trades.

Code	Constante	Description
10004	TRADE_RETCODE_REQUOTE	Recotation (requote)
10006	TRADE_RETCODE_REJECT	Demande rejetée
10007	TRADE_RETCODE_CANCEL	Demande annulée par le trader
10008	TRADE_RETCODE_PLACED	Ordre placé
10009	TRADE_RETCODE_DONE	Demande terminée
10010	TRADE_RETCODE_DONE_PARTIAL	La demande a été complétée partiellement
10011	TRADE_RETCODE_ERROR	Erreur de traitement de la demande
10012	TRADE_RETCODE_TIMEOUT	Demande annulée en raison du délai trop long
10013	TRADE_RETCODE_INVALID	Demande invalide
10014	TRADE_RETCODE_INVALID_VOLUME	Volume invalide dans la demande
10015	TRADE_RETCODE_INVALID_PRICE	Prix invalide dans la demande
10016	TRADE_RETCODE_INVALID_STOPS	Stops invalides dans la demande
10017	TRADE_RETCODE_TRADE_DISABLED	Le trading est désactivé
10018	TRADE_RETCODE_MARKET_CLOSED	Le marché est fermé
10019	TRADE_RETCODE_NO_MONEY	Pas assez d'argent pour compléter la demande
10020	TRADE_RETCODE_PRICE_CHANGED	Les prix ont changé
10021	TRADE_RETCODE_PRICE_OFF	Il n'y a pas de cotation pour traiter la demande
10022	TRADE_RETCODE_INVALID_EXPIRATION	Date d'expiration de l'ordre invalide dans la demande
10023	TRADE_RETCODE_ORDER_CHANGED	L'état de l'ordre a changé
10024	TRADE_RETCODE_TOO_MANY_REQUESTS	Demandes trop fréquentes
10025	TRADE_RETCODE_NO_CHANGES	Aucun changement dans la demande
10026	TRADE_RETCODE_SERVER_DISABLES_AT	Trading automatique désactivé par le serveur

© 2000-2024, MetaQuotes Ltd.

Code	Constante	Description
10027	TRADE_RETCODE_CLIENT_DISABLES_AT	Trading automatique désactivé par le terminal client
10028	TRADE_RETCODE_LOCKED	Demande verrouillée pour traitement
10029	TRADE_RETCODE_FROZEN	Ordre ou position gelé
10030	TRADE_RETCODE_INVALID_FILL	Type de remplissage d'ordre invalide
10031	TRADE_RETCODE_CONNECTION	Aucune connexion au serveur de trades
10032	TRADE_RETCODE_ONLY_REAL	L'opération n'est autorisée que pour les comptes réels
10033	TRADE_RETCODE_LIMIT_ORDERS	Le nombre limite d'ordres en attente est atteint
10034	TRADE_RETCODE_LIMIT_VOLUME	Le volume des ordres et des positions pour le symbole a atteint la limite autorisée
10035	TRADE_RETCODE_INVALID_ORDER	Type d'ordre incorrect ou interdit
10036	TRADE_RETCODE_POSITION_CLOSED	La position avec le POSITION_IDENTIFIER spécifié a déjà été fermée
10038	TRADE_RETCODE_INVALID_CLOSE_VOLUME	Le volume de clôture est supérieur au volume actuel de la position
10039	TRADE_RETCODE_CLOSE_ORDER_EXIST	Un ordre de clôture existe déjà pour la position spécifiée Ceci peut arriver lors de l'utilisation du hedging : • lors de la tentative de clôturer une position avec une position opposée, alors que des ordres de clôture pour la position existent déjà • lors de la tentative de clôturer complétement ou partiellement une position, si le volume total des ordres de clôture existants et du nouvel ordre placé est supérieur au volume actuel de la position
10040	TRADE_RETCODE_LIMIT_POSITIONS	Le nombre de positions ouvertes simultanément sur un compte peut être limité par le paramètrage du serveur. Lorsque la limite est atteinte, le serveur retourne l'erreur TRADE_RETCODE_LIMIT_POSITIONS lors de la demande de placement d'un nouvel ordre. La limite se comporte différemment suivant le type de comptabilisation des positions:

© 2000-2024, MetaQuotes Ltd.

Code	Constante	Description
		• Netting — le nombre de positions ouvertes est pris en compte. Lorsque la limite est atteinte, la plateforme désactive le placement de nouveaux ordres dont l'exécution augmenterait le nombre de positions ouvertes. La plateforme n'autorise en fait que le placement d'ordres pour les symboles ayant déjà des positions ouvertes. Les ordres en attente en cours ne sont pas considérés puisque leur exécution peut amener des changements pour les positions actuelles mais n'augmente pas leur nombre. • Hedging — les ordres en attente sont pris en compte de même que les positions ouvertes, puisque l'activation d'un ordre en attente conduit toujours à ouvrir une nouvelle position. Lorsque la limite est atteinte, la plateforme désactive à la fois les nouveaux ordres au marché pour ouvrir une position et les ordres en attente.
10041	TRADE_RETCODE_REJECT_CANCEL	La demande d'activation de l'ordre en attente est rejetée, l'ordre est annulé
10042	TRADE_RETCODE_LONG_ONLY	La demande est rejetée car le paramètre "**Seules les positions long sont autorisées**" est activé pour le symbole (POSITION_TYPE_BUY)
10043	TRADE_RETCODE_SHORT_ONLY	La demande est rejetée car le paramètre "**Seules les positions short sont autorisées**" est activé pour le symbole (POSITION_TYPE_SELL)
10044	TRADE_RETCODE_CLOSE_ONLY	La demande est rejetée car le paramètre "**Seules les clôtures de positions sont autorisées**" est activé pour le symbole
10045	TRADE_RETCODE_FIFO_CLOSE	La requête est rejetée car le flag "**La fermeture de la position n'est autorisée que par une règle FIFO**" est utilisé pour le compte de trading (ACCOUNT_FIFO_CLOSE=true)
10046	TRADE_RETCODE_HEDGE_PROHIBITED	La requête est rejetée car la règle "**Les positions opposées sur un seul symbole sont désactivées**" est définie pour le compte de trading. Par exemple, si le compte a une position d'achat, un

© 2000-2024, MetaQuotes Ltd.

Code	Constante	Description
		utilisateur ne peut pas ouvrir une position de vente ou passer un ordre de vente en attente. La règle n'est appliquée qu'aux comptes avec système de comptabilité de couverture (ACCOUNT_MARGIN_MODE= ACCOUNT_MARGIN_MODE_RETAIL_HEDGING).

© 2000-2024, MetaQuotes Ltd.

Warnings du Compilateur

Les avertissements (warnings) du compilateurs sont affichés pour information et ne sont pas des messages d'erreur.

Code	Description
21	Données de date incomplètes dans la chaîne de caractères datetime
22	Numéro incorrect dans la chaîne de caractères datetime pour la date. Conditions : Année 1970 <= X <= 3000 Mois 0 <X <= 12 Jour 0 <X <= 31/30/28 (29)….
23	Numéro incorrect dans la chaîne de caractères datetime pour l'heure. Conditions : Heure 0 <= X <24 Minute 0 <= X <60
24	Couleur invalide dans le format RGB : l'une des composantes RGB est inférieure à 0 ou supérieure à 255
25	Caractère inconnu dans les séquences d'échappement. Connus : \n \r \t \\ \" \' \X \x
26	Nombre de variables locales trop grand (> 512Ko) pour la fonction, réduisez leur nombre
29	Enumération déjà définie (duplication) - les membres seront ajoutés à la première définition
30	Surcharge d'une macro
31	La variable est déclarée mais non utilisée
32	Le constructeur doit être du type void
33	Le destructeur doit être du type void
34	La constante ne rentre pas dans l'intervalle des integers (X> _UI64_MAX | | X <_I64_MIN) et sera convertie en type double
35	HEX trop longue - plus de 16 caractères significatifs
36	Pas de valeurs dans la chaîne HEX "0x"
37	Aucune fonction - rien à effectuer
38	Une variable non initialisée est utilisée
41	La fonction n'a pas de corps et n'est pas appelée
43	Perte potentielle de données pendant le casting de type. Exemple : int x = (double) z;
44	Perte de précision des données lors de la conversion de la constante. Exemple : int x = M_PI

© 2000-2024, MetaQuotes Ltd.

Code	Description
45	Différence entre les signes des opérandes dans les opérations de comparaison. Exemple : (char) c1> (uchar) c2
46	Problèmes avec l'import de fonction - la déclaration de #import est requise ou l'import de fonctions est fermé
47	Description trop grande - les caractères supplémentaires ne seront pas inclus dans le fichier exécutable
48	Le nombre de buffers d'indicateurs déclarés est inférieur au nombre nécessaire
49	Aucune couleur pour dessiner une série graphique dans l'indicateur
50	Aucune série graphique pour dessiner l'indicateur
51	Fonction 'OnStart' non trouvée dans le script
52	Fonction 'OnStart' définie avec des paramètres incorrects
53	La fonction 'OnStart' ne peut être définie que dans un script
54	La fonction 'OnInit' est définie avec de mauvais paramètres
55	La fonction 'OnInit' n'est pas utilisée dans les scripts
56	La fonction 'OnDeinit' est définie avec de mauvais paramètres
57	La fonction 'OnDeinit' n'est pas utilisée dans le script
58	Deux fonctions 'OnCalculate' sont définies. OnCalculate() à un seul tableau de prix sera utilisée
59	Surcharge détectée lors du calcul d'un integer constant complexe
60	La variable est peut être non initialisée.
61	Cette déclaration rend impossible la référence à la variable locale déclarée à la ligne spécifiée
62	Cette déclaration rend impossible la référence à la variable globale déclarée à la ligne spécifiée
63	Ne peut pas être utilisé pour un tableau alloué de façon statique
64	Cette déclaration de variable cache une variable prédéfinie
65	La valeur de l'expression est toujours vraie/fausse
66	L'utilisation d'une variable ou d'un expression de type bool dans des opérations mathématiques n'est pas sécurisé
67	Le résultat de l'application de l'opérateur unaire moins à un type non signé ulong est indéfini
68	La version spécifiée dans la propriété #property version n'est pas acceptable pour la section Market ; le format correct de #property version est "XXX.YYY"
69	Déclaration contrôllée vide trouvée

© 2000-2024, MetaQuotes Ltd.

Code	Description
70	Type de retour de fonction invalide ou paramètres incorrects pendant la déclaration de la fonction de gestion de l'évènement
71	Un cast de structures implicite vers un type est requis
72	Cette déclaration rend l'accès direct impossible au membre d'une classe déclaré dans la chaîne de caractères spécifiée. L'accès ne sera possible qu'avec l'opérateur de résolution de portée ::
73	La constante binaire est trop grande, les chiffres du plus grand ordre seront tronqués
74	Le paramètre dans la méthode de la classe héritée a un modificateur const différent, la fonction dérivée a surchargé la fonction parente
75	Valeur de décalage négative ou trop grande dans l'l'opération de décalage de bits, le résultat de l'exécution est non défini
76	La fonction doît retourner une valeur
77	La fonction void retourne une valeur
78	Tous les chemins de contrôle ne retournent pas une valeur
79	Les expressions ne sont pas autorisées à un niveau global
80	Vérifiez la précédence des opérateurs pour de possibles erreurs ; utilisez des parenthèses pour clarifier la précédence
81	Deux fonctions OnCalCulate() sont définies. La version OHLC sera utilisée
82	La struct n'a pas de membres, la taille est définie à 1 octet
83	La valeur de retour de la fonction devrait être vérifiée
84	L'indicateur de ressource est compilé pour debugging. Cela ralentit les performances. Veuillez recompiler l'indicateur pour augmenter les performances
85	Code de caractères trop grand dans la chaîne de caractères, doit être dans l'intervalle 0 à 65535
86	Caractère non reconnu dans la chaîne de caractères
87	Aucun propriété de fenêtre d'indicateur (l'affichage sera fait dans la fenêtre principale ou dans une sous-fenêtre) n'est définie. La propriété #property indicator_chart_window est appliquée

© 2000-2024, MetaQuotes Ltd.

Erreurs de Compilation

MetaEditor 5 affiche les messages des erreurs des programmes détectées pendant la compilation par le compilateur intégré. La liste de ces erreurs est donnée ci-dessous dans le tableau. Pour compile un code source en un fichier exécutable, appuyez sur la touche **F7**. Les programmes qui contiennent des erreurs ne peuvent pas être compilés tant que les erreurs identifiées par le compilateur ne sont pas corrigées.

Code	Description
100	Erreur de lecture du fichier
101	Erreur d'ouverture d'un fichier *. EX5 en écriture
103	Pas assez de mémoire pour terminer la compilation
104	Unité syntaxique vide non reconnue par le compilateur
105	Nom de fichier incorrect dans #include
106	Erreur d'accès au fichier spécifié dans le #include (le fichier n'existe probablement pas)
108	Nom inapproprié pour le #define
109	Commande de préprocesseur inconnue (valider les #include, #define, #property, #import)
110	Symbole inconnu pour le compilateur
111	Fonction non implémentée (la description est présente, mais pas le corps de la fonction)
112	Il manque un guillemet (double quote - ")
113	Il manque un chevron (<) ou un guillemet (")
114	Il manque une apostrophe (single quote - ') omitted
115	Il manque un chevron ">"
116	Type non spécifié dans la déclaration
117	Aucun opérateur de retour, ou return n'existe pas dans toutes les branches de l'implémentation
118	Parenthèse ouvrante attendue
119	Erreur d'écriture du fichier EX5
120	Accès invalide à un tableau
121	La fonction n'est pas de type void et l'opérateur return doit retourner une valeur
122	Déclaration incorrecte du destructeur
123	Symbole ":" manquant
124	Variable déjà déclarée

© 2000-2024, MetaQuotes Ltd.

Code	Description
125	Une variable avec cet identifiant existe déjà
126	Nom de la variable trop long (> 250 caractères)
127	Une structure avec cet identifiant est déjà définie
128	La structure n'est pas définie
129	Un membre de la structure est déja défini avec ce nom
130	Pas de membre de la structure avec ce nom
131	Les parenthèses ne correspondent pas
132	Parenthèse ouvrant "(" attendue
133	Parenthèse fermante manquante
134	Trop difficile à compiler (trop de branchements, niveaux de pile interne sur-remplis)
135	Erreur d'ouverture du fichier en lecture
136	Pas assez de mémoire pour télécharger le fichier source en mémoire
137	Variable attendue
138	La référence ne peut pas être initialisée
140	Assignement attendu (apparaît à la déclaration)
141	Parenthèse ouvrante "{" attendue
142	Le paramètre ne peut être qu'un tableau dynamique
<143	Utilisation impossible du type "void"
144	Parenthèse ou crochet ouvrant manquant
145	Parenthèse ou crochet fermant manquant
146	Taille de tableau incorrecte
147	Trop de paramètres (> 64)
149	Cet élément n'est pas attendu ici
150	Utilisation incorrecte de l'opération (opérandes invalides)
151	Expression de type void non autorisée
152	Opérateur attendu
153	Mauvaise utilisation de break
154	Point virgule ";" attendu
155	Virgule "," attendue
156	Doit être de type class, et non pas struct

© 2000-2024, MetaQuotes Ltd.

Code	Description
157	Expression attendue
158	Caractère non hexadécimal trouvé dans HEX ou nombre trop grand (nombre de chiffres > 511)
159	Chaîne de caractères constante de plus de 65534 caractères
160	Une définition de fonction n'est pas possible ici
161	Fin du programme non attendue
162	Les déclarations forward sont interdites pour les structures
163	Une fonction avec ce nom est déjà définie et a un autre type de retour
164	Une fonction avec ce nom est déjà définie et a un ensemble de paramètres différent
165	Une fonction avec ce nom est déjà définie et est déjà implémentée
166	La surcharge de fonction pour cet appel n'a pas été trouvé
167	Une fonction avec une valeur de retour de type void ne peut pas retourner de valeur
168	Fonction non définie
170	Valeur attendue
171	Dans une expression *case*, seules des constantes de type integer sont valides
172	La valeur du *case* dans ce *switch* est déjà utilisée
173	Integer attendu
174	Dans l'expression #import, un nom de fichier est attendu
175	Les expressions ne sont pas autorisées au niveau global
176	Parenthèse fermante ")" manquante avant ";"
177	Une variable est attendue à gauche du signe d'égalité
178	Le résultat de l'expression n'est pas utilisé
179	Les déclarations de variables ne sont pas autorisées dans un *case*
180	Conversion implicite d'une chaîne de caractères en nombre
181	Conversion implicite d'un nombre en une chaîne de caractères
182	Appel ambigu à une fonction surchargée (plusieurs surcharges conviennent)
183	*else* illégal sans *if* correspondant
184	*case* invalide ou *default* sans *switch*
185	Utilisation inappropriée d'une ellipse
186	La séquence d'initialisation a plus d'éléments que la variable initialisée

© 2000-2024, MetaQuotes Ltd.

Code	Description
187	Constante attendue pour le *case*
188	Expression constante attendue
189	Une variable constante ne peut pas être modifiée
s190	Parenthèse fermante ou virgule attendue (lors de la déclaration d'un membre du tableau)
191	Identifiant d'énumération déjà défini
192	Une énumération ne peut pas avoir de modificateurs d'accès (const, extern, static)
193	Membre d'énumération déjà déclaré avec une valeur différente
194	Une variable avec ce nom est déjà définie
195	Une structure avec ce nom est déjà définie
196	Nom du membre de l'énumération attendu
197	Expression de type integer attendue
198	Division par zéro dans une expression constante
199	Nombre incorrect de paramètres dans la fonction
200	Le paramètre par référence doit être une variable
201	Une variable de même type passée par référence attendue
202	Une variable constante ne peut pas être passée par une référence non constante
203	Nécessite une constante integer positive
204	Echec d'accès au membre protected de la classe
205	Import déjà défini d'une autre façon
208	Fichier exécutable non créé
209	Point d'entrée 'OnCalculate' entry non trouvé pour l'indicateur
210	L'opération continue ne peut être utilisée que dans une boucle
211	Erreur d'accès à un membre privé de classe
213	Méthode de structure ou de classe non déclarée
214	Erreur d'accès à une méthode privée de classe
216	Copie de structures contenant des objets non autorisée
218	Indice en dehors de l'intervalle du tableau
219	Initialisation du tableau dans la déclaration d'une classe ou d'une structure non autorisée
220	Le constructeur de classe ne peut pas avoir de paramètres

© 2000-2024, MetaQuotes Ltd.

Code	Description
221	Un destructeur de classe ne peut pas avoir de paramètres
222	Une méthode de classe ou de structure avec le même nom et les mêmes paramètres est déjà déclarée
223	Opérande attendue
224	Une méthode de class ou de structure avec le même nom existe déjà, mais avec des paramètres différents (déclaration!=implémentation)
225	Fonction importée non décrite
226	ZeroMemory() n'est pas autorisée pour les objets ayant des membres protected ou ayant de l'héritage
227	Appel ambigu à la surcharge de fonction (correspondance exacte des paramètres pour plusieurs surcharges)
228	Nom de variable attendu
229	Une référence ne peut pas être déclarée ici
230	Déjà utilisé comme nom de l'énumération
232	Classe ou structure attendue
235	Ne peut appeler l'opérateur 'delete' pour supprimer le tableau
236	Opérateur ' while' attendu
237	L'opérateur 'delete' doit avoir un pointeur
238	'default' déjà existant pour ce 'switch'
239	Erreur de syntaxe
240	Une séquence d'échappement ne peut apparaître que dans une chaîne de caractères (commence avec '\')
241	Tableau requis - le crochet ouvrant '[' ne s'applique pas à un tableau, ou autre chose qu'un tableau est passé comme paramètre au tableau
242	Ne peut pas être initialisé avec la liste d'initialisation
243	Import non défini
244	Erreur d'optimisation sur l'arbre syntaxique
245	Trop de structures déclarées (essayez de simplifier le programme)
246	Conversion du paramètre non autorisée
247	Utilisation incorrecte de l'opérateur 'delete'
248	Il n'est pas autorisé de déclarer un pointeur sur une référence
249	Il n'est pas autorisé de déclarer une référence sur une référence
250	Il n'est pas autorisé de déclarer un pointeur sur un pointeur

© 2000-2024, MetaQuotes Ltd.

Code	Description
251	La déclaration de structure dans la liste des paramètres n'est pas autorisée
252	Opération de casting de type invalide
253	Un pointeur ne peut être déclaré que pour une classe ou une structure
256	Identifiant non déclaré
257	Erreur d'optimisation du code exécutable
258	Erreur de génération du code exécutable
260	Expression invalide pour l'opérateur 'switch'
261	Pool de constantes de type chaînes de caractères rempli, simplifiez le programme
262	Impossible de convertir vers une énumération
263	Ne pas utiliser 'virtual' pour les données (membres de classe ou de structure)
264	Ne peut pas appeler une méthode protected de classe
265	Fonctions virtuelles surchargées avec des types de retour différents
266	Une classe ne peut pas dériver d'une structure
267	Une structure ne peut pas dériver d'une classe
268	Un constructeur ne peut être virtuel (le spécificateur *virtual* n'est pas autorisé)
269	Une méthode d'une structure ne peut pas être virtuelle
270	La fonction doit avoir un corps
271	La surcharge de fonctions système (fonctions du terminal) n'est pas autorisée
272	Spécificateur *Const* invalide pour les fonctions non membres d'une classe ou d'une structure
274	Impossible de changer un membre de classe dans une méthode constante
276	Séquence d'initialisation inappropriée
277	Valeur par défaut manquante pour le paramètre (déclaration spécifique des paramètres par défaut)
278	Surcharge du paramètre par défaut (différentes valeurs dans la déclaration et dans l'implémentation)
279	Il n'est pas autorisé d'appeler une méthode non constante pour un objet constant
280	Un objet est nécessaire pour accéder aux membres
281	Le nom d'une structure déjà déclarée ne peut pas être utilisé dans la déclaration
284	Conversion non autorisée (héritage fermé)
285	Les structures et tableaux ne peuvent pas être utilisés comme variables d'entrée
286	Le spécificateur *const* n'est pas valide pour les constructeurs/destructeurs

© 2000-2024, MetaQuotes Ltd.

Code	Description
287	Expression de chaîne de caractères incorrecte pour une datetime
288	Propriété inconnue (#property)
289	Valeur de propriété incorrecte
290	Indice de propriété invalide dans #property
291	Paramètre d'appel manquant - <func (x,)>
293	L'objet doit être passé par référence
294	Le tableau doit être passé par référence
295	La fonction est déclarée comme exportable
296	La fonction n'est pas déclarée comme exportable
297	Il est interdit d'exporter une fonction importée
298	Les fonctions importées ne peuvent pas avoir ce paramètre (il est interdit de passer un pointeur, une classe ou une structure contenant un tableau dynamique, un pointeur, une classe, etc.)
299	Doit être une classe
300	#import non fermé
302	Les types ne correspondent pas
303	Variable externe déjà initialisée
304	Aucune fonction exportée ou point d'entrée trouvé
305	L'appel à un constructeur explicite n'est pas autorisé
306	Méthode déclarée comme constante
307	Méthode non déclarée comme constante
308	Taille incorrecte du fichier de ressource
309	Nom de ressource incorrect
310	Erreur d'ouverture du fichier de ressource
311	Erreur de lecture du fichier de ressource
312	Type de ressource inconnu
313	Chemin vers le fichier de ressorce incorrect
314	Le nom spécifié de la ressource est déjà utilisé
315	Argument attendu pour la macro
316	Symbole inattendu dans la définition de la macro
317	Erreur dans les paramètres de la macro

© 2000-2024, MetaQuotes Ltd.

Code	Description
318	Nombre invalide de paramètres pour la macro
319	Trop de paramètres pour la macro
320	Trop complexe, simplifiez la macro
321	Le paramètre pour EnumToString() ne peut être qu'une énumération
322	Le nom de la ressource est trop long
323	Format d'image non supporté (seul les BMP en 24 ou 32 bits sont supportées)
324	Un tableau ne peut être déclaré dans un opérateur
325	La fonction ne peut être déclarée qu'au niveau global
326	La déclaration n'est pas autorisée dans la portée actuelle
327	L'initialisation de variables statiques avec les valeurs de variables locales n'est pas autorisée
328	Déclaration illégale d'un tableau d'objets n'ayant pas de constructeur par défaut
329	Les listes d'initialisation ne sont autorisées que dans les constructeurs
330	Pas de définition de fonction avec la liste d'initialisation
331	Liste d'initialisation vide
332	L'initialisation d'un tableau dans un constructeur n'est pas autorisée
333	L'initialisation de membres d'une classe parente dans la liste d'initialisation n'est pas autorisée
334	Expression de type integer type attendue
335	La mémoire requise pour le tableau excède la valeur maximum
336	La mémoire requise pour la structure excède la valeur maximum
337	La mémoire requise pour les variables déclarées au niveau global excède la valeur maximum
338	La mémoire requise pour les variables locales excède la valeur maximum
339	Constructeur non défini
340	Nom invalide pour le fichier d'icone
341	Impossible d'ouvrir le fichier d'iconeavec le chemin spécifié
342	Le fichier d'icone est incorrect est n'est pas au format ICO
343	Réinitialisation d'un membre dans le constructeur de la classe/structure par la liste d'initialisation
344	L'initialisation de membres statiques dans la liste d'initialisation du constructeur n'est pas autorisée

© 2000-2024, MetaQuotes Ltd.

Code	Description
345	L'initialisation d'un membre non-statique d'une classe/structure à un niveau global n'est pas autorisée
346	Le nom de la méthode de la classe/structure correspond au nom d'un membre déclaré auparavant
347	Le nom d'un membre de la classe/structure correspond au nom d'une méthode déclarée auparavant
348	Une fonction virtuelle ne peut pas être déclarée comme statique
349	Le modificateur const n'est pas autorisé pour les fonctions statiques
350	Un constructeur ou un destructeur ne peut pas être statique
351	Un membre ou une méthode non statique d'une classe ou d'une structure ne peut pas être accédée depuis une fonction statique
352	Une surcharge (+,-,[],++,--, etc.) est attendue après le mot-clé operator
353	Toutes les opérations ne peuvent pas être surchargées en MQL5
354	La définition ne correspond pas à la déclaration
355	Un nombre invalide de paramètres est spécifié pour l'opérateur
356	Fonction de gestion d'évènement non trouvée
357	La méthode ne peut pas être exportée
358	Un pointeur sur un objet constant ne peut pas être normalisé par un objet non constant
359	Les templates de classe ne sont pas encore supportés
360	La surcharge d'un template de fonction n'est pas encore supporté
361	Le template de fonction ne peut pas être appliqué
362	Paramètre ambigu dans le template de fonction (plusieurs types de paramètres peuvent être appliqués)
363	Impossible de déterminer le type du paramètre avec lequel l'argument de la fonction templatée devrait être appelée
364	Nombre incorrect de paramètres dans le template de la fonction
365	Un template de fonction ne peut pas être virtuel
366	Les templates de fonctions ne peuvent pas être exportés
367	Les templates de fonctions ne peuvent pas être importés
368	Les structures contenant des objets ne sont pas autorisées
369	Les tableaux de chaînes de caractères et les structures contenant des objets ne sont pas autorisés

© 2000-2024, MetaQuotes Ltd.

Code	Description
370	Un membre statique de classe/structure doit être initialisé explicitement
371	Limitation du compilateur : une chaîne de caractères ne peut pas contenir plus de 65 535 caractères
372	#ifdef/#endif inconsistents
373	Un objet de cette classe ne peut pas être retourné car aucun constructeur par copie n'existe
374	Les membres et méthodes non statiques ne peuvent pas être utilisés
375	Impossible d'utiliser OnTesterInit() sans OnTesterDeinit()
376	Redéfinition du paramètre formel '%s'
377	Les macros __FUNCSIG__ et __FUNCTION__ ne peuvent pas apparaître en dehors du corps d'une fonction
378	Type de retour invalide Par exemple, cette erreur sera produite pour les fonctions importées de DLL retournant une structure ou un pointeur.
379	Erreur d'utilisation du template
380	Non utilisé
381	Syntaxe illégale lors de la déclaration d'une fonction virtuelle pure, seuls "=NULL" ou "=0" sont autorisés
382	Seules les fonctions virtuelles peuvent être déclarées avec le spécificateur pure ("=NULL" ou "=0")
383	Une classe abstraite ne peut pas être instanciée
384	Un pointeur sur un type utilisateur devrait être mis comme type de destination pour un cast dynamique en utilisant l'opérateur dynamic_cast
385	Un type "pointeur sur fonction" est attendu
386	Les pointeurs sur méthodes ne sont pas supportés
387	Erreur - ne peut pas définir le type d'un pointeur sur une fonction
388	La cast de type n'est pas possible en raison de l'héritage privé
389	Une variable avec le modificateur const doit être initialisée pendant la déclaration
393	Seules les méthodes avec un accès public peuvent être déclarées dans une interface
394	Imbrication invalide d'une interface dans une autre interface
395	Une interface ne peut dériver que d'une autre interface
396	Une interface est attendue
397	Les interfaces ne supportent que l'héritage public
398	Une interface ne peut pas contenir de membres

© 2000-2024, MetaQuotes Ltd.

Code	Description
399	Les objets interface ne peuvent pas être créés directement, ils ne peuvent qu'utiliser l'héritage
400	Un spécificateur ne peut pas être utilisé dans une déclaration de type forward
401	Héritage de la classe impossible, puisqu'elle est déclarée avec le spécificateur final
402	Ne peut pas redéfinir une méthode déclarée avec le spécificateur final
403	Le spécificateur final ne peut être appliqué qu'aux fonctions virtuelles
404	La méthode marquée avec le spécificateur override ne surcharge aucune fonction de la classe de base
405	Un spécificateur n'est pas autorisé dans la définition d'une fonction, seulement dans sa déclaration
406	Ne peut pas convertir le type dans le type demandé
407	Le type ne peut pas être utilisé pour une variable ressource
408	Erreur dans le fichier projet
409	Ne peut pas être utilisé comme membre d'une union
410	Choix ambigü pour le nom, le contexte d'usage doit être défini explicitement
411	La structure ne peut pas être utilisée depuis une DLL
412	Ne peut pas appeler une fonction marquée avec le spécificateur delete
413	MQL4 non pris en charge. Pour compiler le programme, utilisez MetaEditor dans le dossier d'installation du terminal MetaTrader 4

© 2000-2024, MetaQuotes Ltd.

Erreurs d'Exécution (runtime)

GetLastError() est la fonction retournant le dernier code d'erreur stocké dans la variable prédéfinie _LastError. Cette valeur peut être remise à zéro avec la fonction ResetLastError().

Constante	Code	Description
ERR_SUCCESS	0	Opération terminée avec succès
ERR_INTERNAL_ERROR	4001	Erreur interne inattendue
ERR_WRONG_INTERNAL_PARAMETER	4002	Paramètre incorrect dans l'appel interne à la fonction du terminal client
ERR_INVALID_PARAMETER	4003	Paramètre incorrect dans l'appel à la fonction système
ERR_NOT_ENOUGH_MEMORY	4004	Pas assez de mémoire pour effectuer la fonction système
ERR_STRUCT_WITHOBJECTS_ORCLASS	4005	La structure contient des objets de type string et/ou des tableaux dynamiques et/ou une structure de tels objets et/ou classes
ERR_INVALID_ARRAY	4006	Tableau de type ou de taille invalide, ou l'objet du tableau dynamique est endommagé
ERR_ARRAY_RESIZE_ERROR	4007	Pas assez de mémoire pour la réallocation d'un tableau, ou pour changer la taille d'un tableau statique
ERR_STRING_RESIZE_ERROR	4008	Pas assez de mémoire pour la réallocation d'une chaîne de caractères
ERR_NOTINITIALIZED_STRING	4009	Chaîne de caractère non initialisée
ERR_INVALID_DATETIME	4010	Date et/ou heure invalide
ERR_ARRAY_BAD_SIZE	4011	Le nombre total d'éléments du tableau ne peut pas excéder 2 147 483 647
ERR_INVALID_POINTER	4012	Pointeur incorrect
ERR_INVALID_POINTER_TYPE	4013	Type de pointeur incorrect
ERR_FUNCTION_NOT_ALLOWED	4014	Appel à la fonction non autorisé
ERR_RESOURCE_NAME_DUPLICATED	4015	Les noms des ressources dynamique et statique sont identiques

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
ERR_RESOURCE_NOT_FOUND	4016	Aucune ressource trouvée avec ce nom dans EX5
ERR_RESOURCE_UNSUPPORTED_TYPE	4017	Type de ressource non supportée ou sa taille est supérieure à 16 Mo
ERR_RESOURCE_NAME_IS_TOO_LONG	4018	Le nom de la ressource excède 63 caractères
ERR_MATH_OVERFLOW	4019	Dépassement mémoire (overflow) lors du calcul de la fonction mathématique
ERR_SLEEP_ERROR	4020	Date de fin de test en dehors de l'appel après avoir appelé Sleep()
ERR_PROGRAM_STOPPED	4022	Test arrêté de force de l'extérieur. Par exemple, optimisation interrompue, fenêtre de test visuel fermée ou agent de test arrêté
Graphiques		
ERR_CHART_WRONG_ID	4101	Identifiant de graphique incorrect
ERR_CHART_NO_REPLY	4102	Le graphique ne répond pas
ERR_CHART_NOT_FOUND	4103	Graphique non trouvé
ERR_CHART_NO_EXPERT	4104	Aucun Expert Advisor pouvant gérer l'évènement n'est présent sur le graphique
ERR_CHART_CANNOT_OPEN	4105	Erreur d'ouverture du graphique
ERR_CHART_CANNOT_CHANGE	4106	Echec de changement du symbole ou de la période du graphique
ERR_CHART_WRONG_PARAMETER	4107	Valeur d'erreur du paramètre des fonctions pour travailler avec les graphiques
ERR_CHART_CANNOT_CREATE_TIMER	4108	Echec de création du timer
ERR_CHART_WRONG_PROPERTY	4109	Identifiant de propriété du graphique incorrect
ERR_CHART_SCREENSHOT_FAILED	4110	Erreur de création de la copie d'écran
ERR_CHART_NAVIGATE_FAILED	4111	Erreur de navigation dans le graphique
ERR_CHART_TEMPLATE_FAILED	4112	Erreur lors de l'application du modèle

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
ERR_CHART_WINDOW_NOT_FOUND	4113	La sous-fenêtre contenant l'indicateur n'a pas été trouvée
ERR_CHART_INDICATOR_CANNOT_ADD	4114	Erreur lors de l'ajout de l'indicateur sur le graphique
ERR_CHART_INDICATOR_CANNOT_DEL	4115	Erreur de suppression d'un indicateur du graphique
ERR_CHART_INDICATOR_NOT_FOUND	4116	Indicateur non trouvé sur le graphique spécifié
Objets graphiques		
ERR_OBJECT_ERROR	4201	Erreur lors de l'utilisation d'un objet graphique
ERR_OBJECT_NOT_FOUND	4202	L'objet graphique n'a pas été trouvé
ERR_OBJECT_WRONG_PROPERTY	4203	Identifiant de propriété d'un objet graphique incorrect
ERR_OBJECT_GETDATE_FAILED	4204	Impossible de récupérer la date correspondant à la valeur
ERR_OBJECT_GETVALUE_FAILED	4205	Impossible de récupérer la valeur correspondant à la date
MarketInfo		
ERR_MARKET_UNKNOWN_SYMBOL	4301	Symbole inconnu
ERR_MARKET_NOT_SELECTED	4302	Le symbole n'est pas sélectionné dans le MarketWatch
ERR_MARKET_WRONG_PROPERTY	4303	Identifiant de propriété du symbole incorrect
ERR_MARKET_LASTTIME_UNKNOWN	4304	L'heure du dernier tick est inconnue (aucun tick)
ERR_MARKET_SELECT_ERROR	4305	Erreur lors de l'ajout ou de la suppression d'un symbole dans le MarketWatch
Accès à l'Historique		
ERR_HISTORY_NOT_FOUND	4401	Historique demandé non trouvé
ERR_HISTORY_WRONG_PROPERTY	4402	Identifiant de propriété d'historique incorrect
ERR_HISTORY_TIMEOUT	4403	Demande d'historique expirée (timeout)

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
ERR_HISTORY_BARS_LIMIT	4404	Nombre de barres demandées limité par les paramètres du terminal
ERR_HISTORY_LOAD_ERRORS	4405	Multiples erreurs lors du chargement de l'historique
ERR_HISTORY_SMALL_BUFFER	4407	Tableau de destination trop petit pour stocker toutes les données demandées
Variables Globales		
ERR_GLOBALVARIABLE_NOT_FOUND	4501	Variable globale du terminal client non trouvée
ERR_GLOBALVARIABLE_EXISTS	4502	Une variable globale du terminal client existe déjà avec ce nom
ERR_GLOBALVARIABLE_NOT_MODIFIED	4503	Variable globale non modifiée
ERR_GLOBALVARIABLE_CANNOTREAD	4504	Erreur de lecture du fichier des valeurs des variables globales
ERR_GLOBALVARIABLE_CANNOTWRITE	4505	Erreur d'écriture du fichier des valeurs des variables globales
ERR_PLAY_SOUND_FAILED	4511	Echec de lecture du son
ERR_MQL5_WRONG_PROPERTY	4512	Identifiant de propriété du programme incorrect
ERR_TERMINAL_WRONG_PROPERTY	4513	Identifiant de propriété du terminal incorrect
ERR_FTP_SEND_FAILED	4514	Echec de l'envoi d'un fichier en ftp
ERR_NOTIFICATION_SEND_FAILED	4515	Echec d'envoi d'une notification
ERR_NOTIFICATION_WRONG_PARAMETER	4516	Paramètre invalide pour l'envoi d'une notification - une chaîne de caractères vide ou NULL a été envoyé à la fonction SendNotification()
ERR_NOTIFICATION_WRONG_SETTINGS	4517	Paramètre incorrect des notifications dans le terminal (l'identifiant n'est pas spécifié ou la permission n'est pas définie)
ERR_NOTIFICATION_TOO_FREQUENT	4518	Envois de notifications trop fréquents
ERR_FTP_NOSERVER	4519	Serveur FTP non spécifié
ERR_FTP_NOLOGIN	4520	Login FTP non spécifié

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
ERR_FTP_FILE_ERROR	4521	Fichier à envoyer sur le serveur FTP non trouvé dans le répertoire MQL5\Files
ERR_FTP_CONNECT_FAILED	4522	Echec de la connexion FTP
ERR_FTP_CHANGEDIR	4523	Chemin FTP non trouvé sur le serveur
Buffers des Indicateurs Personnalisés		
ERR_BUFFERS_NO_MEMORY	4601	Pas assez de mémoire pour les buffers des indicateurs
ERR_BUFFERS_WRONG_INDEX	4602	Indice incorrect du buffer de l'indicateur
Propriétés des Indicateurs Personnalisés		
ERR_CUSTOM_WRONG_PROPERTY	4603	Identifiant de propriété de l'indicateur personnalisé incorrect
Compte		
ERR_ACCOUNT_WRONG_PROPERTY	4701	Identifiant de propriété du compte incorrect
ERR_TRADE_WRONG_PROPERTY	4751	Identifiant de propriété de trade incorrect
ERR_TRADE_DISABLED	4752	Trading par les Expert Advisors interdit
ERR_TRADE_POSITION_NOT_FOUND	4753	Position non trouvée
ERR_TRADE_ORDER_NOT_FOUND	4754	Ordre non trouvé
ERR_TRADE_DEAL_NOT_FOUND	4755	Transaction non trouvée
ERR_TRADE_SEND_FAILED	4756	Echec de l'envoi de la demande de trade
ERR_TRADE_CALC_FAILED	4758	Echec du calcul du profit ou de la marge
Indicateurs		
ERR_INDICATOR_UNKNOWN_SYMBOL	4801	Symbole inconnu
ERR_INDICATOR_CANNOT_CREATE	4802	L'indicateur ne peut pas être créé
ERR_INDICATOR_NO_MEMORY	4803	Pas assez de mémoire pour ajouter l'indicateur
ERR_INDICATOR_CANNOT_APPLY	4804	L'indicateur ne peut pas être appliqué à un autre indicateur

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
ERR_INDICATOR_CANNOT_ADD	4805	Erreur lors de l'application d'un indicateur sur le graphique
ERR_INDICATOR_DATA_NOT_FOUND	4806	Données demandées non trouvées
ERR_INDICATOR_WRONG_HANDLE	4807	Handle d'indicateur incorrect
ERR_INDICATOR_WRONG_PARAMETERS	4808	Nombre de paramètres incorrect lors de la création d'un indicateur
ERR_INDICATOR_PARAMETERS_MISSING	4809	Aucun paramètre lors de la création d'un indicateur
ERR_INDICATOR_CUSTOM_NAME	4810	Le premier paramètre du tableau doit être le nom de l'indicateur personnalisé
ERR_INDICATOR_PARAMETER_TYPE	4811	Type de paramètre invalide dans le tableau lors de la création d'un indicateur
ERR_INDICATOR_WRONG_INDEX	4812	Indice incorrect du buffer demandé de l'indicateur
Depth of Market		
ERR_BOOKS_CANNOT_ADD	4901	Le Depth Of Market ne peut pas être ajouté
ERR_BOOKS_CANNOT_DELETE	4902	Le Depth Of Market ne peut pas être enlevé
ERR_BOOKS_CANNOT_GET	4903	Impossible d'obtenir les données du Depth Of Market
ERR_BOOKS_CANNOT_SUBSCRIBE	4904	Erreur lors de l'abonneemnt pour reçevoir les nouvelles données du Depth Of Market
Opérations sur les Fichiers		
ERR_TOO_MANY_FILES	5001	Impossible d'ouvrir plus de 64 fichiers en même temps
ERR_WRONG_FILENAME	5002	Nom de fichier invalide
ERR_TOO_LONG_FILENAME	5003	Nom de fichier trop long
ERR_CANNOT_OPEN_FILE	5004	Erreur d'ouverture du fichier
ERR_FILE_CACHEBUFFER_ERROR	5005	Pas assez de mémoire en cache pour lire le fichier
ERR_CANNOT_DELETE_FILE	5006	Erreur de suppression du fichier
ERR_INVALID_FILEHANDLE	5007	Un fichier avec ce handle a déjà été fermé, ou n'a pas été ouvert

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
ERR_WRONG_FILEHANDLE	5008	Handle de fichier incorrect
ERR_FILE_NOTTOWRITE	5009	Le fichier doit être ouvert en écriture
ERR_FILE_NOTTOREAD	5010	Le fichier doit être ouvert en lecture
ERR_FILE_NOTBIN	5011	Le fichier doit être ouvert en mode binaire
ERR_FILE_NOTTXT	5012	Le fichier doit être ouvert en mode texte
ERR_FILE_NOTTXTORCSV	5013	Le fichier doit être ouvert en mode texte ou CSV
ERR_FILE_NOTCSV	5014	Le fichier doit être ouvert en mode CSV
ERR_FILE_READERROR	5015	Erreur de lecture du fichier
ERR_FILE_BINSTRINGSIZE	5016	La taille de la chaîne de caractères doit être spécifiée, car le fichier est ouvert en mode binaire
ERR_INCOMPATIBLE_FILE	5017	Un fichier texte est prévu pour les tableaux de chaînes de caractères, pour les autres tableaux, utiliser le mode binaire
ERR_FILE_IS_DIRECTORY	5018	Ce n'est pas un fichier mais un répertoire
ERR_FILE_NOT_EXIST	5019	Le fichier n'existe pas
ERR_FILE_CANNOT_REWRITE	5020	Impossible d'écrire dans le fichier
ERR_WRONG_DIRECTORYNAME	5021	Nom de répertoire incorrect
ERR_DIRECTORY_NOT_EXIST	5022	Le répertoire n'existe pas
ERR_FILE_ISNOT_DIRECTORY	5023	Ce n'est pas un répertoire mais un fichier
ERR_CANNOT_DELETE_DIRECTORY	5024	Le répertoire ne peut pas être supprimé
ERR_CANNOT_CLEAN_DIRECTORY	5025	Echec de suppression du répertoire (un ou plusieurs fichiers sont peut-être bloqués et l'opération de suppression a échoué)

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
ERR_FILE_WRITEERROR	5026	Echec d'écriture de la ressource vers un fichier
ERR_FILE_ENDOFFILE	5027	Imposible de lire l'élément suivant du fichier CSV (FileReadString, FileReadNumber, FileReadDatetime, FileReadBool), car la fin du fichier est atteinte
Cast des Chaînes de Caractères		
ERR_NO_STRING_DATE	5030	Aucune date dans la chaîne de caractères
ERR_WRONG_STRING_DATE	5031	Date incorrecte dans la chaîne de caractères
ERR_WRONG_STRING_TIME	5032	Heure incorrecte dans la chaîne de caractères
ERR_STRING_TIME_ERROR	5033	Erreur de conversion de la chaîne de caractères en date
ERR_STRING_OUT_OF_MEMORY	5034	Pas assez de mémoire pour la chaîne de caractères
ERR_STRING_SMALL_LEN	5035	La longueur de la chaîne de caractères est inférieure à la taille attendue
ERR_STRING_TOO_BIGNUMBER	5036	Nombre trop grand, supérieur à ULONG_MAX
ERR_WRONG_FORMATSTRING	5037	Formatage de la chaîne de caractères invalide
ERR_TOO_MANY_FORMATTERS	5038	Nombre de spécificateurs de format supérieur au nombre de paramètres
ERR_TOO_MANY_PARAMETERS	5039	Nombre de paramètres supérieur au nombre de spécificateurs de format
ERR_WRONG_STRING_PARAMETER	5040	Paramètre de type string endommagé
ERR_STRINGPOS_OUTOFRANGE	5041	Position en dehors de la chaîne de caractères
ERR_STRING_ZEROADDED	5042	0 ajouté à la fin de la chaîne de caractères, opération inutile
ERR_STRING_UNKNOWNTYPE	5043	Type de données inconnu lors de la conversion vers une chaîne de

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
		caractères
ERR_WRONG_STRING_OBJECT	5044	Objet string endommagé
Opérations sur les tableaux		
ERR_INCOMPATIBLE_ARRAYS	5050	Copie de tableaux incompatibles Le tableau de chaîne de caractères ne peut être copié que vers un tableau de chaîne de caractèresn et un tableau numérique vers un tableau numérique
ERR_SMALL_ASSERIES_ARRAY	5051	Le tableau destination est déclaré comme AS_SERIES, et n'a pas une taille suffisante
ERR_SMALL_ARRAY	5052	Tableau trop petit, le position de départ est en dehors du tableau
ERR_ZEROSIZE_ARRAY	5053	Tableau de longueur zéro
ERR_NUMBER_ARRAYS_ONLY	5054	Doit être un tableau numérique
ERR_ONEDIM_ARRAYS_ONLY	5055	Doit être un tableau à 1 dimension
ERR_SERIES_ARRAY	5056	Timeseries ne peut pas être utilisé
ERR_DOUBLE_ARRAY_ONLY	5057	Doit être un tableau de type double
ERR_FLOAT_ARRAY_ONLY	5058	Doit être un tableau de type float
ERR_LONG_ARRAY_ONLY	5059	Doit être un tableau de type long
ERR_INT_ARRAY_ONLY	5060	Doit être un tableau de type int
ERR_SHORT_ARRAY_ONLY	5061	Doit être un tableau de type short
ERR_CHAR_ARRAY_ONLY	5062	Doit être un tableau de type char
ERR_STRING_ARRAY_ONLY	5063	Doit être un tableau de type char
Opérations avec OpenCL		
ERR_OPENCL_NOT_SUPPORTED	5100	Les fonctions OpenCL ne sont pas supportées sur cet ordinateur
ERR_OPENCL_INTERNAL	5101	Une erreur interne est survenue lors de l'exécution d'OpenCL
ERR_OPENCL_INVALID_HANDLE	5102	Handle OpenCL invalide

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
ERR_OPENCL_CONTEXT_CREATE	5103	Erreur lors de la création du contexte OpenCL
ERR_OPENCL_QUEUE_CREATE	5104	Echec de création d'une queue dans OpenCL
ERR_OPENCL_PROGRAM_CREATE	5105	Une erreur est survenue lors de la compilation d'un programme OpenCL
ERR_OPENCL_TOO_LONG_KERNEL_NAME	5106	Nom de noyau (kernel) trop long (noyau OpenCL)
ERR_OPENCL_KERNEL_CREATE	5107	Erreur de création d'un noyau OpenCL
ERR_OPENCL_SET_KERNEL_PARAMETER	5108	Une erreur est survenue lors de la définition des paramètres pour le noyau OpenCL
ERR_OPENCL_EXECUTE	5109	Erreur d'exécution du programme OpenCL runtime error
ERR_OPENCL_WRONG_BUFFER_SIZE	5110	Taille invalide du buffer OpenCL
ERR_OPENCL_WRONG_BUFFER_OFFSET	5111	Offset invalide du buffer OpenCL
ERR_OPENCL_BUFFER_CREATE	5112	Echec de création d'un buffer OpenCL
ERR_OPENCL_TOO_MANY_OBJECTS	5113	Trop d'objets OpenCL
ERR_OPENCL_SELECTDEVICE	5114	Erreur de sélection du périphérique OpenCL
Working with databases		
ERR_DATABASE_INTERNAL	5120	Erreur interne de la base de données
ERR_DATABASE_INVALID_HANDLE	5121	Handle incorrect de base de données
ERR_DATABASE_TOO_MANY_OBJECTS	5122	Nombre maximum d'objets de base de données acceptables atteint
ERR_DATABASE_CONNECT	5123	Erreur de connexion à la base de données
ERR_DATABASE_EXECUTE	5124	Erreur d'exécution de la requête
ERR_DATABASE_PREPARE	5125	Erreur de génération de la requête
ERR_DATABASE_NO_MORE_DATA	5126	Plus de données à lire

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
ERR_DATABASE_STEP	5127	Echec du déplacement sur l'entrée suivante de la requête
ERR_DATABASE_NOT_READY	5128	Les données pour la lecture des résultats de la requête ne sont pas encore prêts
ERR_DATABASE_BIND_PARAMETERS	5129	Echec de la substitution des paramètress de la requête SQL
Opérations avec WebRequest		
ERR_WEBREQUEST_INVALID_ADDRESS	5200	URL invalide
ERR_WEBREQUEST_CONNECT_FAILED	5201	Echec de connexion à l'URL spécifiée
ERR_WEBREQUEST_TIMEOUT	5202	Délai dépassé
ERR_WEBREQUEST_REQUEST_FAILED	5203	Echec de la requête HTTP
Opérations réseau (sockets)		
ERR_NETSOCKET_INVALIDHANDLE	5270	Handle de socket invalide passé à la fonction
ERR_NETSOCKET_TOO_MANY_OPENED	5271	Trop de sockets ouvertes (max 128)
ERR_NETSOCKET_CANNOT_CONNECT	5272	Echec de connexion à l'hôte distant
ERR_NETSOCKET_IO_ERROR	5273	Echec de l'envoi ou de la réception de données depuis la socket
ERR_NETSOCKET_HANDSHAKE_FAILED	5274	Echec d'établissement de la connexion sécurisée (TLS Handshake)
ERR_NETSOCKET_NO_CERTIFICATE	5275	Aucune donnée concernant le certificat protégeant la connexion
Symboles Personnalisés		
ERR_NOT_CUSTOM_SYMBOL	5300	Un symbole personnalisé doit être spécifié
ERR_CUSTOM_SYMBOL_WRONG_NAME	5301	Le nom du symbole personnalisé est invalide. Le nom du symbole ne peut contenir que des lettres latines, sans ponctuation, espace ou caractère spécial (ne peut contenir que ".", "_", "&" and "#"). Il n'est pas recommandé d'utiliser

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
		les caractères <, >, :, ", /,\, |, ?, *.
ERR_CUSTOM_SYMBOL_NAME_LONG	5302	Le nom du symbole personnalisé est trop long. La longueur du nom du symbole ne doit pas être supérieure à 32 caractères, incluant le caractère 0 terminal
ERR_CUSTOM_SYMBOL_PATH_LONG	5303	Le chemin du symbole personnalisé est trop long. La longueur du chemin ne doit pas excéder 128 caractères, incluant "Custom\\", le nom du symbole, les séparateurs de groupes et le 0 terminal
ERR_CUSTOM_SYMBOL_EXIST	5304	Un symbole personnalisé existe déjà avec ce nom
ERR_CUSTOM_SYMBOL_ERROR	5305	Une erreur est survenue lors de la création, de la suppression ou du changement du symbole personnalisé
ERR_CUSTOM_SYMBOL_SELECTED	5306	Vous essayez de supprimer un symbole personnalisé sélectionné dans le Market Watch
ERR_CUSTOM_SYMBOL_PROPERTY_WRONG	5307	Propriété invalide du symbole personnalisé
ERR_CUSTOM_SYMBOL_PARAMETER_ERROR	5308	Paramètre incorrect lors de la définition de la propriété d'un symbole personnalisé
ERR_CUSTOM_SYMBOL_PARAMETER_LONG	5309	Paramètre de type string trop long lors de la définition de la propriété d'un symbole personnalisé
ERR_CUSTOM_TICKS_WRONG_ORDER	5310	Les ticks du tableau ne sont pas ordonnés par heure
Calendrier Economique		
ERR_CALENDAR_MORE_DATA	5400	La taille du tableau est insuffisante pour recevoir les descriptions de toutes les valeurs
ERR_CALENDAR_TIMEOUT	5401	Délai d'attente de la requête dépassé
ERR_CALENDAR_NO_DATA	5402	Pays non trouvé

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
Utilisation des bases de données		
ERR_DATABASE_ERROR	5601	Erreur générique
ERR_DATABASE_LOGIC	5602	Erreur de logique interne de SQLite
ERR_DATABASE_PERM	5603	Accés refusé
ERR_DATABASE_ABORT	5604	Annulation de l'appel à la callback
ERR_DATABASE_BUSY	5605	Fichier de la base de données verrouillé
ERR_DATABASE_LOCKED	5606	Table de la base de données verrouillée
ERR_DATABASE_NOMEM	5607	Mémoire insuffisante pour terminer l'opération
ERR_DATABASE_READONLY	5608	Tentative d'écriture dans une base de données en lecture seule
ERR_DATABASE_INTERRUPT	5609	Opération terminée par sqlite3_interrupt()
ERR_DATABASE_IOERR	5610	Erreur de lecture/écriture sur le disque
ERR_DATABASE_CORRUPT	5611	Image disque de la base de données corrompue
ERR_DATABASE_NOTFOUND	5612	Code d'opération inconnu dans sqlite3_file_control()
ERR_DATABASE_FULL	5613	Echec de l'insertion car la base de données est pleine
ERR_DATABASE_CANTOPEN	5614	Impossible d'ouvrir le fichier de la base de données
ERR_DATABASE_PROTOCOL	5615	Erreur de protocole de verrouillage de la base de données
ERR_DATABASE_EMPTY	5616	Usage interne uniquement
ERR_DATABASE_SCHEMA	5617	Schéma modifié de la base de données
ERR_DATABASE_TOOBIG	5618	La taille de la chaîne de caractères ou du BLOB est supérieure à la limite autorisée
ERR_DATABASE_CONSTRAINT	5619	Annulation suite à la violation d'une contrainte
ERR_DATABASE_MISMATCH	5620	Différence de type des données

© 2000-2024, MetaQuotes Ltd.

Constante	Code	Description
ERR_DATABASE_MISUSE	5621	Bibliothèque utilisée incorrectement
ERR_DATABASE_NOLFS	5622	Utilisation des fonctionnalités du système d'exploitation non supportée sur l'hôte
ERR_DATABASE_AUTH	5623	Autorisation refusée
ERR_DATABASE_FORMAT	5624	Non utilisé
ERR_DATABASE_RANGE	5625	Erreur de liaison du paramètre, indice incorrect
ERR_DATABASE_NOTADB	5626	Le fichier ouvert n'est pas un fichier de base de données
Erreurs Personnalisées		
ERR_USER_ERROR_FIRST	65536	Les erreurs personnalisées commencent avec ce code

Voir aussi

Codes de Retour du Serveur de Trading

© 2000-2024, MetaQuotes Ltd.

Constantes d'Entrée et de Sortie

Constantes :

- Flags d'ouverture de fichier
- Propriétés des Fichiers
- Se positionner dans un fichier
- Utilisation d'une page de code
- MessageBox

© 2000-2024, MetaQuotes Ltd.

Flags d'Ouverture de Fichier

Les valeurs des flags d'ouverture de fichiers spécifient le mode d'accès au fichier. Les flags sont définis comme suit :

Identificateur	Valeur	Description
FILE_READ	1	Le fichier est ouvert en lecture. Le flag est utilisé par FileOpen(). Lors de l'ouverture d'un fichier, il est nécessaire de spécifier FILE_WRITE et/ou FILE_READ.
FILE_WRITE	2	Le fichier est ouvert en écriture. Le flag est utilisé par FileOpen(). Lors de l'ouverture d'un fichier, il est nécessaire de spécifier FILE_WRITE et/ou FILE_READ.
FILE_BIN	4	Mode de lecture/écriture binaire (sans conversion en chaîne de caractères). Le flag est utilisé par FileOpen().
FILE_CSV	8	Fichier CSV (tous ses éléments sont convertis en chaînes de caractères du type correspondant, Unicode ou ANSI, et séparé par un séparateur). Le flag est utilisé par FileOpen().
FILE_TXT	16	Fichier texte simple (similaire au fichier csv, mais sans utiliser de séparateur). Le flag est utilisé par FileOpen().
FILE_ANSI	32	Chaînes de caractères de type ANSI (symboles de 1 octet). Le flag est utilisé par FileOpen().
FILE_UNICODE	64	Chaînes de caractères de type UNICODE (2 octets par symbole). Le flag est utilisé par FileOpen().
FILE_SHARE_READ	128	Accès partagé en lecture depuis plusieurs programmes. Le flag es utilisé dans FileOpen(), mais il ne remplace pas la nécessité d'indiquer les flags FILE_WRITE et/ou FILE_READ lors de l'ouverture du fichier.
FILE_SHARE_WRITE	256	Accès partagé en écriture pour plusieurs programmes. Le flag es utilisé dans FileOpen(), mais il ne remplace pas la nécessité d'indiquer les flags FILE_WRITE et/ou FILE_READ lors de l'ouverture du fichier.
FILE_REWRITE	512	Possibilité de réécriture du fichier avec les fonctions FileCopy() et FileMove(). Le fichier doit exister ou doit être ouvert en écriture, sinon le fichier ne sera pas ouvert.
FILE_COMMON	4096	Le chemin relatif au répertoire commun à tous les terminaux clients \Terminal\Common\Files. Le flag est utilisé dans les fonctions FileOpen(), FileCopy(), FileMove() et FileIsExist().

Un ou plusieurs fichiers peuvent être spécifiés lors de l'ouverture d'un fichier. Les flags peuvent être combinés. Les flags sont combinés avec le signe du OU logique (|), qui est positionné entre les flags

© 2000-2024, MetaQuotes Ltd.

énumérés. Par exemple, pour ouvrir un fichier au format CSV en lecture-écriture, spécifiez la combinaison FILE_READ|FILE_WRITE|FILE_CSV.

Exemple :

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

Certaines spécifités sont à savoir lors de l'utilisation des flags de lecture/écriture :

- Si FILE_READ est spécifié, une tentative d'ouverture d'un fichier existant est effectuée. Si le fichier n'existe pas, l'ouverture du fichier échoue et aucun nouveau fichier n'est créé.
- FILE_READ|FILE_WRITE – un nouveau fichier est créé si aucun fichier n'existe avec le nom spécifié.
- FILE_WRITE – le fichier est recréé avec une taille à zéro.

Lors de l'ouverture d'un fichier, il est nécessaire de spécifier FILE_WRITE et/ou FILE_READ.

Les flags définissant le type de lecture d'un fichier ouvert ont des priorités. Le flag avec la plus haute priorité est FILE_CSV, puis FILE_BIN, et FILE_TXT qui a la priorité la plus basse. Si plusieurs flags sont spécifiés en même temps (FILE_TXT|FILE_CSV ou FILE_TXT|FILE_BIN ou FILE_BIN|FILE_CSV), le flag ayant la plus haute priorité est donc utilisé.

Les flags définissant le type d'encodage ont également des priorités. FILE_UNICODE a une priorité plus haute que FILE_ANSI. Si vous spécifiez une combinaison FILE_UNICODE|FILE_ANSI, le flag FILE_UNICODE sera utilisé.

Si ni FILE_UNICODE ni FILE_ANSI ne sont spécifiés, FILE_UNICODE est implicite. Si ni FILE_CSV, ni FILE_BIN, ni FILE_TXT ne sont spécifiés, FILE_CSV est implicite.

Si un fichier est ouvert en lecture en mode fichier texte (FILE_TXT ou FILE_CSV), et que les deux octets 0xff,0xfe sont trouvés au début du fichier, le flag d'encodage sera FILE_UNICODE, même si FILE_ANSI est spécifié.

Voir aussi

Fonctions sur les Fichiers

© 2000-2024, MetaQuotes Ltd.

Propriétés des Fichiers

La fonction FileGetInteger() est utilisée pour obtenir les propriétés d'un fichier. L'identifiant de la propriété désirée de l'énumération ENUM_FILE_PROPERTY_INTEGER est passé lors de l'appel.

ENUM_FILE_PROPERTY_INTEGER

Identifiant	Description
FILE_EXISTS	Vérifie l'existence
FILE_CREATE_DATE	Date de création
FILE_MODIFY_DATE	Date de la dernière modification
FILE_ACCESS_DATE	Date du dernière accès au fichier
FILE_SIZE	Taille du fichier en octets
FILE_POSITION	Position d'un pointeur dans le fichier
FILE_END	Récupère le signe de fin de fichier
FILE_LINE_END	Récupère le signe de début de fichier
FILE_IS_COMMON	Le fichier est ouvert dans un répertoire partagé par tous les terminaux (voir FILE_COMMON)
FILE_IS_TEXT	Le fichier est ouvert en mode fichier texte (voir FILE_TXT)
FILE_IS_BINARY	Le fichier est ouvert en mode fichier binaire (voir FILE_BIN)
FILE_IS_CSV	Le fichier est ouvert en mode fichier CSV (voir FILE_CSV)
FILE_IS_ANSI	Le fichier est ouvert en mode fichier ANSI (voir FILE_ANSI)
FILE_IS_READABLE	Le fichier est ouvert en lecture (voir FILE_READ)
FILE_IS_WRITABLE	Le fichier est ouvert en écriture (voir FILE_WRITE)

La fonction FileGetInteger() peut être appelée de deux façons différentes. Dans la première option, pour récupérer les propriétés d'un fichier, son handle, qui est obtenu lors de l'ouverture du fichier avec la fonction FileOpen(), est spécifié. Cette option permet de récupérer toutes les propriétés du fichier.

La deuxième option de la fonction FileGetInteger() retourne les valeurs des propriétés du fichier par son nom. Avec cette option, seules les propriétés générales suivantes peuvent être obtenues :

- FILE_EXISTS - existence d'un fichier ayant le nom spécifié
- FILE_CREATE_DATE - date de création du fichier avec le nom spécifié
- FILE_MODIFY_DATE - date de modification du fichier avec le nom spécifié
- FILE_ACCESS_DATE - date du dernier accès au fichier avec le nom spécifié
- FILE_SIZE - taille du fichier avec le nom spécifié

Lorsque l'on tente de récupérer des propriétés autres que celles spécifiées ci-dessus, la deuxième option de l'appel à FileGetInteger() retournera une erreur.

© 2000-2024, MetaQuotes Ltd.

Se Positionner dans un Fichier

La plupart des fonctions sur les fichiers sont associées aux opérations de lecture/écriture des donnée. En même type, en utilisnt la fonction FileSeek(), vous pouvez spécifier la position d'un pointeur de fichier dans le fichier, à partir de laquelle l'opération de lecture ou d'écriture suivante sera effectué. L'énumération ENUM_FILE_POSITION contient les positions valides du pointeur, relativement auxquelles vous pouvez spécifier le décalage en octets pour l'opération suivante.

ENUM_FILE_POSITION

Identificateur	Description
SEEK_SET	Début du fichier
SEEK_CUR	Position actuelle d'un pointeur de fichier
SEEK_END	Fin du fichier

Voir aussi

FileIsEnding, FileIsLineEnding

© 2000-2024, MetaQuotes Ltd.

Utilisation d'une Page de Code dans les Opérations de Conversion des Chaînes de Caractères

Lors de la conversion de variables de type string en tableaux de char et inversement, l'encodage correspondant par défaut à l'ANSI du système d'opération Windows (CP_ACP) est utilisé en MQL5. Si vous souhaitez spécifier un différent type d'encodage, li peut être défini comme paramètre additionnel pour les fonctions CharArrayToString(), StringToCharArray() et FileOpen() functions.

Le tableau liste les constantes intégrées pour quelques unes des pages de code les plus populaires. Les pages de code non mentionnées peuvent être spécifiées par le code correspondant à la page.

Constantes et Codes de Page intégrés

Constante	Valeur	Description
CP_ACP	0	La page de code Windows ANSI actuelle.
CP_OEMCP	1	La page de code du système OEM actuel.
CP_MACCP	2	La page de code du système Macintosh actuel. **Note :** Cette valeur est principalement utilisé dans les anciens programmes et n'est plus utilisé aujourd'hui, puisque les ordinateurs Macintosh récents utilisent l'Unicode pour l'encodage.
CP_THREAD_ACP	3	La page de code Windows ANSI pour le thread actuel.
CP_SYMBOL	42	Page de code des symboles
CP_UTF7	65000	Page de code UTF-7.
CP_UTF8	65001	Page de code UTF-8.

Voir aussi

Propriétés du Terminal Client

© 2000-2024, MetaQuotes Ltd.

Constantes de la boîte de dialogue MessageBox

Cette section contient les codes de retour de la fonction MessageBox() function. Si une fenêtre de message a un bouton Annuler, la fonction retourne IDCANCEL si la touche Echap ou le bouton Annuler ont été utilisé. S'il n'y a pas de bouton Annuler dans la fenêtre de message, un appui sur la touche Echap n'a aucun effet.

Constante	Valeur	Description
IDOK	1	Le bouton "OK" a été appuyé
IDCANCEL	2	Le bouton "Annuler" (Cancel) a été appuyé
IDABORT	3	Le bouton "Annuler" (Abort) a été appuyé
IDRETRY	4	Le bouton "Réessayer" (Retry) a été appuyé
IDIGNORE	5	Le bouton "Ignorer" a été utilisé
IDYES	6	Le bouton "Oui" a été appuyé
IDNO	7	Le bouton "Non" a été appuyé
IDTRYAGAIN	10	Le bouton "Réessayer" (Try again) a été appuyé
IDCONTINUE	11	Le bouton "Continuer" a été appuyé

Les principaux flags de la fonction MessageBox() définissent le contenu et le comportement de la fenêtre du message. Cette valeur peut être une combinaison des groupes de flags suivants :

Constante	Valeur	Description
MB_OK	0x00000000	Le fenêtre de message ne contient qu'un seul bouton : OK. Par défaut
MB_OKCANCEL	0x00000001	Le fenêtre de message contient deux boutons : OK et Annuler (Cancel).
MB_ABORTRETRYIGNORE	0x00000002	Le fenêtre de message contient trois bouton : Annuler (Abort), Réessayer (Retry) et Ignorer.
MB_YESNOCANCEL	0x00000003	Le fenêtre de message contient trois bouton : Oui, Non et Annuler (Cancel).
MB_YESNO	0x00000004	Le fenêtre de message contient deux boutons : Oui et Non.
MB_RETRYCANCEL	0x00000005	Le fenêtre de message contient deux boutons : Réessayer (Retry) et Annuler (Cancel).
MB_CANCELTRYCONTINUE	0x00000006	Le fenêtre de message contient trois bouton : Réessayer (Try again) et Continuer.

© 2000-2024, MetaQuotes Ltd.

Pour afficher une icone dans la fenêtre de message, il est nécessaire de spécifier deux flags supplémentaires :

Constante	Valeur	Description
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	0x00000010	L'icone du signe STOP
MB_ICONQUESTION	0x00000020	L'icone du signe Question
MB_ICONEXCLAMATION, MB_ICONWARNING	0x00000030	L'icone du signe Exclamation/Avertissement
MB_ICONINFORMATION, MB_ICONASTERISK	0x00000040	Le sign **i** dans un cercle (information)

Les boutons par défaut sont définis avec les flags suivants :

Constante	Valeur	Description
MB_DEFBUTTON1	0x00000000	Le premier bouton MB_DEFBUTTON1 - est le bouton par défaut, si les autres boutons MB_DEFBUTTON2, MB_DEFBUTTON3, ou MB_DEFBUTTON4 ne sont pas spécifiés
MB_DEFBUTTON2	0x00000100	Le second bouton est le bouton par défaut
MB_DEFBUTTON3	0x00000200	Le troisième bouton est le bouton par défaut
MB_DEFBUTTON4	0x00000300	Le quatrième bouton est le bouton par défaut

© 2000-2024, MetaQuotes Ltd.

Les programmes MQL5

Pour que le programme mql5 puisse fonctionner elle doit être compilée (le bouton "Compiler" ou la touche F7). La compilation doit passer sans erreurs (les préventions sont admissibles, il faut les analyser). En cela dans le répertoire correspondant, *terminal_dir\MQL5\Experts*, *terminal_dir\MQL5\indicators* ou *terminal_dir\MQL5\scripts*, doit être créé un fichier exécuté avec le même nom et l'extension EX5. Justement ce fichier peut être lancé sur l'exécution.

Les particularités du travail des programmes mql5 sont décrites dans les paragraphes:

- L'exécution des programmes – l'ordre de l'appel des fonctions prédéterminées - des gestionnaires des événements;
- Le test des stratégies commerciales – les particularités du travail des programmes dans le testeur des stratégies;
- Les événements du terminal de client – la description des événements, dont on peut traiter dans les programmes;
- L'appel des fonctions importées – les ordres de la description, les paramètres admissibles, l'ordre de la recherche et l'accord sur les liens des fonctions importées;
- Les erreurs de l'exécution – la réception de l'information sur les erreurs de l'exécution et les erreurs critiques.

Les experts, les indicateurs d'utilisateur et les scripts se fixent à un des graphiques ouverts par la méthode Drag'n'Drop de la fenêtre "le Navigateur" du terminal de client sur le graphique correspondant (la technologie Drag'n'Drop). Les programmes mql5 peuvent fonctionner seulement au terminal de client branché.

Pour que l'expert cesse de travailler, il est nécessaire de le supprimer du graphique en sélectionnant "Les Conseillers - Supprimer" du menu contextuel du graphique. L'état du bouton "Permettre/interdire les conseillers" influence au travail du conseiller.

Pour que l'indicateur d'utilisateur cesse le travail, il est nécessaire de le supprimer du graphique.

Les indicateurs d'utilisateur et les conseillers travaillent jusqu'à ce qu'ils soient explicitement supprimés d'un graphique; l'information sur les conseillers attachés et les indicateurs d'utilisateur est sauvée entre les lancements du terminal de client.

Les scripts sont exécutés une fois et se suppriment automatiquement à la fin du travail ou de la clôture ou le changement de l'état du graphique courant ou à la fin du travail du terminal de client. A la relance du terminal de client les scripts ne sont pas lancés, puisque l'information sur eux n'est pas sauvée.

Maximum un expert, un script et un nombre illimité d'indicateurs peuvent travailler dans un graphique.

Les services n'ont pas besoin d'être liés à un graphique pour fonctionner et sont conçus pour effectuer des opérations supplémentaires. Par exemple, vous pouvez créer dans un service un symbole personnalisé, ouvrir son graphique, obtenir des données dans une boucle infinie avec les fonctions réseau et le mettre à jour en continu.

© 2000-2024, MetaQuotes Ltd.

L'exécution des programmes

Chaque script et chaque expert travaille dans son propre thread séparé. Tous les indicateurs calculés sur un symbole travaillent dans un thread, même s'ils sont lancés sur les graphiques différents. Ainsi tous les indicateurs sur un symbole divisent entre eux-mêmes les ressources d'un thread.

Aussi dans un thread avec les indicateurs on accomplit successivement les autres actions du symbole donné - le traitement des ticks et la synchronisation de l'histoire. Cela signifie que si l'action infinie est accomplie dans l'indicateur, tous les autres événements selon son symbole ne seront jamais accomplis.

Au lancement de l'expert il faut assurer à lui environnement commercialactuel, l'accessibilité de l'histoire selon le symbole donné et la période, ainsi que produire la synchronisation entre le terminal et le serveur. Sur ces procédures le terminal accorde le délai du lancement à l'expert pas plus qu'à 5 secondes, après quoi l'expert sera lancé avec ces données qu'on a réussi à préparer. Voilà pourquoi en absence de communication avec le serveur, ça peut retarder le lancement de l'expert.

Un bref résumé selon les programmes MQL5 est amené dans un tableau:

Programme	Exécution	Note
Script	Dans son propre thread, tant de scripts existent - autant de threads d'exécution il y a pour eux.	Le script avec une boucle infinie ne peut pas perturber le travail des autres programmes
Expert	Dans son propre thread, tant d'experts existent - autant d'experts d'exécution il y a pour eux.	L'expert avec une boucle infinie ne peut pas perturber le travail des autres programmes
Indicateur	Le thread de l'exécution pour tous les indicateurs sur un symbole. Tant de symboles avec indicateurs existent - autant de threads d'exécution il y a pour eux	Une boucle infinie dans un indicateur arrêtera le travail de tous les autres indicateurs sur ce symbole

Directement après qu'un programme est attaché à un graphique on produit son chargement à la mémoire du terminal de client et l'initialisation des variables globales. Si quelque variable globale comme la classe a le constructeur, cet constructeur sera appellé en train de l'initialisation des variables globales.

Après cela le programme se trouve dans l'état de l'attente de l'événement du terminal de client. Chaque programme mql5 avoir être quand même une fonction-gestionnaire de l'événement, dans le cas contraire programme chargé ne sera pas accompli. Les fonctions-gestionnaires des événements ont les noms prédéterminés, les ensembles prédéterminés des paramètres et les types prédéterminés du retour.

© 2000-2024, MetaQuotes Ltd.

Type	Nom de la fonction	Paramètres	Application	Commentaires
int	OnInit	non	Experts et Indicateurs	Le gestionnaire de l'événement Init. On admet le type de la valeur rendue void.
void	OnDeinit	const int reason	Experts et Indicateurs	Le gestionnaire de l'événement Deinit.
void	OnStart	non	Scripts et Services	Le gestionnaire de l'événement Start.
int	OnCalculate	const int rates_total, const int prev_calculated, const datetime &Time[], const double &Open[], const double &High[], const double &Low[], const double &Close[], const long &TickVolume[], const long &Volume[], const int &Spread[]	Indicateurs	Le gestionnaire de l'événement Calculate sur toutes les données de prix.
int	OnCalculate	const int rates_total, const int prev_calculated, const int begin, const double &price[]	Indicateurs	Le gestionnaire de l'événement Calculate sur un tableau des données. Dans l'indicateur ne peut pas simultanément assister 2 gestionnaires Calculate. Dans ce cas travaillera seulement un gestionnaire de l'événement Calculate sur un tableau des données.
void	OnTick	non	Experts	Le gestionnaire de l'événement NewTick. Pendant le traitement de l'événement de l'arrivée du nouveau tick les autres événements de ce type ne viennent pas.
void	OnTimer	non	Experts et	Le gestionnaire de l'événement Timer.

© 2000-2024, MetaQuotes Ltd.

Ty-pe	Nom de la fonction	Paramètres	Application	Commentaires
			Indicateurs	
void	OnTrade	non	Experts	Le gestionnaire de l'événement Trade.
void	OnTester	non	Experts	Le gestionnaire de l'événement Tester.
void	OnChartEvent	const int id, const long &lparam, const double &dparam, const string &sparam	Experts et Indicateurs	Le gestionnaire de l'événement ChartEvent.
void	OnBookEvent	const string &symbol_name	Experts et Indicateurs	Le gestionnaire de l'événement BookEvent.

Le terminal de client envoie les événements apparaissant aux graphiques correspondants ouverts. Aussi les événements peuvent être générés par les graphiques (les événements du graphique) ou par les programmes mql5 (les événements d'utilisateur). On peut activer et désactiver la génération des événements de la création et de la suppression des objets graphiques sur le graphique en spécifiant des propriétés du graphique CHART_EVENT_OBJECT_CREATE et CHART_EVENT_OBJECT_DELETE. Chaque programme MQL5 et chaque graphique a sa propre file d'attente d'événements, où sont ajoutés tous les nouveaux événements entrants.

Le programme reçoit les événements seulement du graphique, où il est lancé. Tous les événements sont traités l'un après l'autre dans l'ordre reçu. Si l'événement NewTick se trouve déjà dans la file d'attente ou cet événement est en état du traitement, on ne met pas un nouvel événement NewTick dans la file d'attente du programme mql5. Analogiquement, si l'événement ChartEventest déjà dans la file d'attente du programme mql5 ou cet événement est en état du traitement, on ne met pas un nouvel événement de ce type dans la file d'attente. Le traitement des événements de la minuterie est produit selon le même schéma – si l'événement Timerest au tour ou est déjà traité, un nouvel événement de la minuterie n'est pas mis au tour.

Les tours des événements ont la taille limitée, mais suffisante, c'est pourquoi le débordement pour les programmes correctement écrits est peu probable. Au débordement du tour les nouveaux événements sont rejetés sans mise dans la file d'attente.

Il n'est pas recommandé d'utiliser des boucles infinies pour traiter les événements. Seulement les scripts qui traitent un seul événement Start peuvent être l'exception de cette règle.

Les bibliothèques ne traitent pas aucuns événements.

L'interdiction à l'utilisation des fonctions dans les indicateurs et les experts

© 2000-2024, MetaQuotes Ltd.

Les indicateurs, les scripts et les experts sont les programmes exécutés de MQL5 et sont destinés aux différents types des tâches. C'est pourquoi il y a une limitations sur l'utilisation des fonctions définies en fonction de type du programme. Les fonctions suivantes sont interdites dans les indicateurs:

- OrderCalcMargin();
- OrderCalcProfit();
- OrderCheck();
- OrderSend();
- SendFTP();
- Sleep();
- ExpertRemove();
- MessageBox().

A son tour, toutes les fonctions, destinées aux indicateurs sont interdites dans les experts et les scripts:

- SetIndexBuffer();
- IndicatorSetDouble();
- IndicatorSetInteger();
- IndicatorSetString();
- PlotIndexSetDouble();
- PlotIndexSetInteger();
- PlotIndexSetString();
- PlotIndexGetInteger.

La bibliothèque n'est pas le programme indépendant et elle s'exécute dans le contexte du programme MQL5 qui l'a appelée: le script, l'indicateur ou l'expert. En conséquence, toutes les limitations indiquées ci-dessus se répandent à la bibliothèque appelée.

Les fonctions sont interdites dans les services

Les services n'acceptent pas d'évènements puisqu'ils ne sont pas liés à un graphique. Les fonctions suivantes sont interdites dans les services :

ExpertRemove();

EventSetMillisecondTimer();

EventSetTimer();

EventKillTimer();

SetIndexBuffer();

IndicatorSetDouble();

IndicatorSetInteger();

IndicatorSetString();

PlotIndexSetDouble();

© 2000-2024, MetaQuotes Ltd.

PlotIndexSetInteger();

PlotIndexSetString();

PlotIndexGetInteger();

Le chargement et le déchargement des indicateurs

Les indicateurs sont chargés dans les cas suivants :

- l'indicateur est attaché à un graphique;
- le lancement du terminal (si l'indicateur était attaché au graphique avant la clôture du terminal);
- le chargement d'un modèle (si l'indicateur attaché à un graphique est spécifié dans le modèle);
- le changement d'un profil (si l'indicateur est attaché à un des graphiques de profil);
- le changement d'un symbole et/ou la période du graphique, auquel l'indicateur est attaché;
- le changement du compte, auquel le terminal est connecté;
- après la recompilation réussie d'un indicateur, si cet indicateur était attaché à un graphique.
- le changement des paramètres d'entrée de l'indicateur.

Les indicateurs sont déchargés dans les cas suivants :

- en détachant un indicateur du graphique;
- la clôture du terminal (si l'indicateur était attaché au graphique);
- le chargement d'un modèle, si un indicateur est attaché au graphique ;
- la fermeture d'un graphique, auquel l'indicateur a été attaché;
- le changement d'un profil, si l'indicateur est attaché à un de graphiques du profil changé;
- le changement d'un symbole et/ou la période du graphique, auquel l'indicateur est attaché;
- le changement du compte, auquel le terminal est connecté;
- le changement les paramètres d'entrée de l'indicateur.

Le chargement et le déchargement des experts

L'expert est chargé dans les cas suivants:

- un expert est attaché à un graphique;
- le lancement du terminal (si l'expert était attaché au graphique avant la clôture du terminal);
- le chargement d'un modèle (si l'expert attaché au graphique est spécifié dans le modèle);
- après la recompilation réussie d'un expert, si cet expert était attaché à un graphique;
- le changement d'un profil, si l'expert est attaché à un de graphiques du profil changé);
- la connexion au compte, même si le numéro du compte n'a pas changé (si l'expert a été fixé au graphique avant l'autorisation du terminal sur le serveur).

Le déchargement de l'expert attaché au graphique, est produit dans les cas suivants:

- en détachant un expert du graphique;

© 2000-2024, MetaQuotes Ltd.

- si un nouvel Expert est attaché à un graphique, si un autre expert a été déjà attaché, cet expert est déchargé;
- la clôture du terminal (si l'expert était attaché au graphique);
- le chargement d'un modèle, si un expert est attaché au graphique;
- la clôture d'un graphique, auquel l'expert est attaché;
- le changement d'un profil, si l'expert est attaché à un de graphiques du profil changé;
- le changement du compte, auquel le terminal est connecté (si l'expert a été fixé au graphique avant l'autorisation du terminal sur le serveur);
- calling the ExpertRemove() function.

Au changement du caractère ou du temps trame du graphique, auquel l'expert est attaché le déchargement et le chargement de l'expert n'est pas produit. Dans ce cas les gestionnaires OnDeinit() sont appelés successivement sur un ancien symbole/temps trame et OnInit() au nouveau symbole/temps trame (s'ils existent), les valeurs des variables globales et des variables statiques ne sont pas réinitialisées. Tous les événements qui sont entrés pour l'expert jusqu'à l'achèvement de l'initialisation (les fonctions OnInit()) se sont manqués.

Le chargement et le déchargement des scripts

Les scripts sont chargés immédiatement après qu'ils sont attachés à un graphique et déchargé immédiatement après qu'ils accomplissent leur opération. En cela les fonctions OnInit() et OnDeinit() ne sont pas appelées pour les scripts.

Au déchargement du programme (supprimer le programme du graphique) se passe la déinitialisation des variables globales et l'annulation du tour des messages. Dans ce cas la déinitialisation signifie la désallocation des variables du type string, la libération des objets des tableaux dynamiques et l'appel des destructeurs à leur présence.

Chargement et Déchargement des services

Les services sont chargés directement après le lancement du terminal s'ils n'étaient pas en cours d'exécution au moment de la fermeture du terminal. Les services ne sont pas déchargés immédiatement après la fin de leur utilisation.

Les services ont un gestionnaire unique OnStart(), dans lequel vous pouvez implémenter une boucle infinie de réception et de traitement des données, par exemple pour créer et mettre à jour des symboles personnalisés en utilisant les fonctions réseau.

Contrairement aux Expert Advisors, indicateurs et scripts, les services ne sont pas liés à un graphique spécifique, un mécanisme spécifique est donc fourni pour les lancer. Une instance d'un nouveau service est créée dans le Navigateur en utilisant la commande "Ajouter un Service". Une instance de service peut être lancée, stoppée et supprimée en utilisant le menu correspondant de l'instance. Pour gérer toutes les instances, utilisez le menu du service.

© 2000-2024, MetaQuotes Ltd.

Pour la meilleure compréhension du travail de l'expert il est recommandé de compiler le code de l'expert amené dans l'exemple et produire les actions du chargement/déchargement des experts, le remplacement de la modèle, le caractère, le temps trame etc.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   TestExpert.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

class CTestClass
  {
public:
   CTestClass() { Print("CTestClass constructor"); }
   ~CTestClass() { Print("CTestClass destructor"); }
  };
CTestClass global;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Initialisation");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print("Deinitialisation with reason ",reason);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
```

Les scripts sont chargés directement après avoir fixé le graphique et déchargé immédiatement après la fin de ses travaux.

Voir aussi

Les événements du terminal de client, Les fonctions de traitement des événements

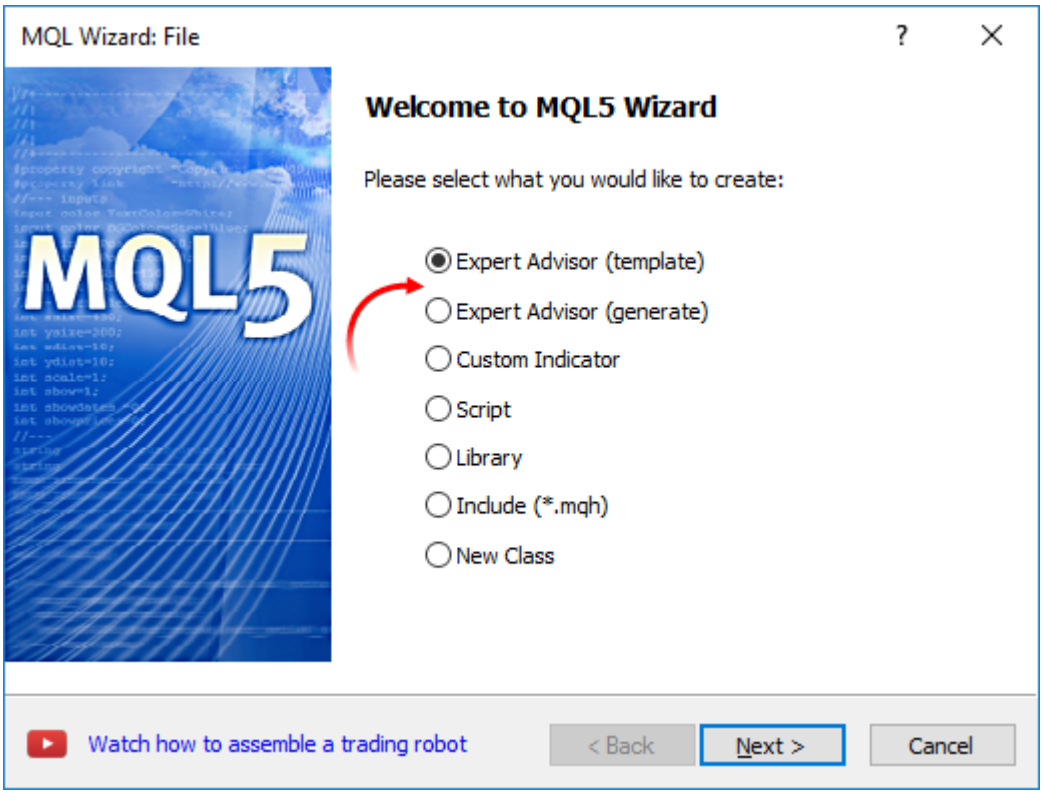
© 2000-2024, MetaQuotes Ltd.

Autorisation de Trading

Automatisation du Trading

Le language MQL5 fournit un groupe spécial de fonctions conçues pour développer des systèmes de trading automatisés. Les programmes élaborés pour le trading automatisé sans aucune intervention humaine sont appelés Expert Advisors ou robots de trading. Pour créer un Expert Advisor dans MetaEditor, lancez l'Assistant MQL5 et sélectionnez l'une des deux options :

- Expert Advisor (template) – vous permet de créer un modèle avec des fonctions de gestion des événements toutes faites qui doivent être complétées par toutes les fonctionnalités nécessaires au moyen de la programmation.
- Expert Advisor (generate) – vous permet de développer un robot de trading à part entièreen sélectionnant simplement les modules nécessaires : module de signaux de trading, module de money management et trailing stop.



Les fonctions de trading ne peuvent travailler qu'avec les Expert Advisors et les scripts. Le trading n'est pas autorisé pour les indicateurs.

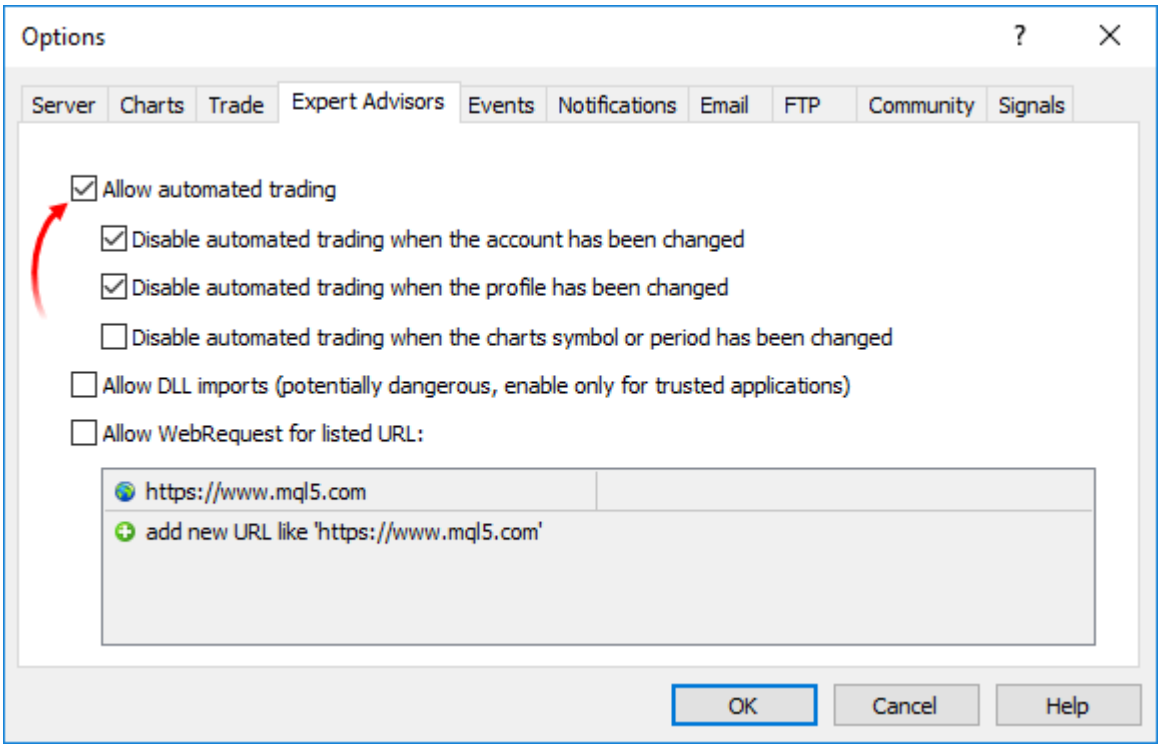
Vérification de l'Autorisation d'Effectuer un Trading Automatisé

Afin de développer un Expert Advisor fiable, capable de travailler sans intervention humaine, il est important de disposer d'un ensemble de contrôles. Premièrement, nous devons vérifier si le programme de trading est autorisé à tous. C'est une vérification de base qui est indispensable lors de l'élaboration d'un système automatisé.

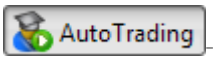
Vérifier l'autorisation d'effectuer le trading automatisé dans le terminal

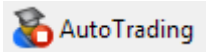
© 2000-2024, MetaQuotes Ltd.

Les paramètres du terminal vous permettent ou non un trading automatisé pour tous les programmes.



Vous pouvez changer l'option du trading automatisé à droite sur le panneau standard du terminal :

- 

- trading automatisé activé, les fonctions de trading dans les applications lancées sont autorisées à être utilisées.
- 

- trading automatisé désactivé, les applications en cours d'exécution sont incapables d'exécuter des fonctions de trading.

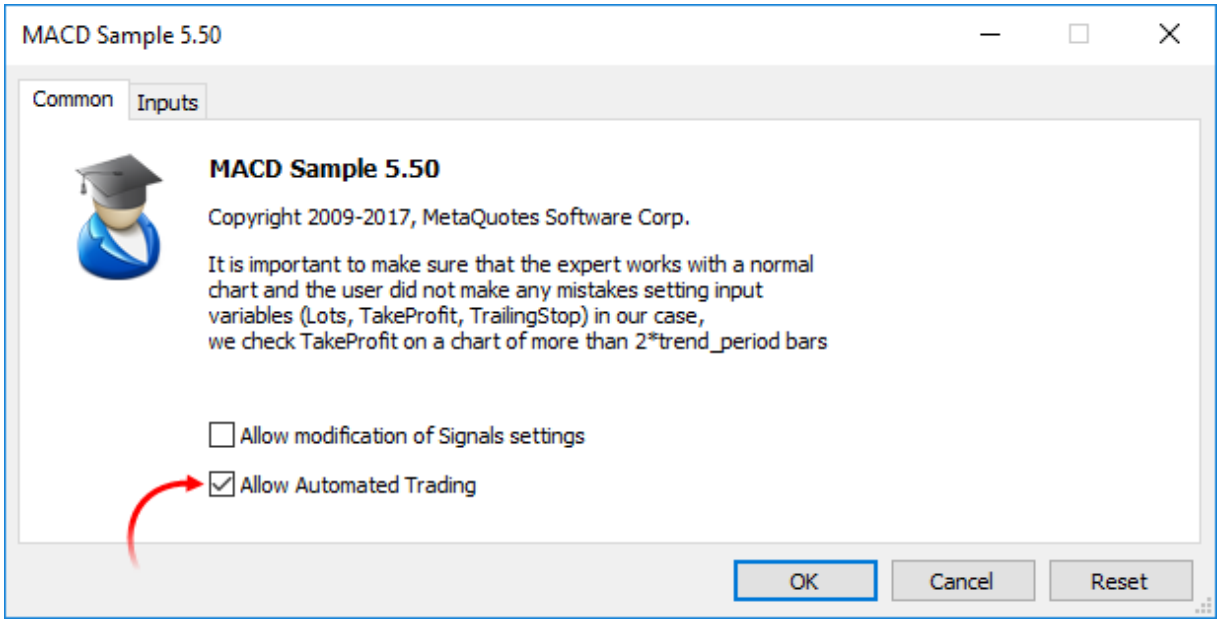
Vérification de l'échantillon :

```
if (!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alerte(" Vérifiez si le trading automatisé est autorisé dans les paramètres du terr
```

Vérifier si le trading est autorisé pour un certain Expert Advisor/script

Vous pouvez autoriser ou interdire le trading automatisé pour un certain programme lors de son lancement. Pour ce faire, utilisez la case à cocher spécifique dans les propriétés du programme.

© 2000-2024, MetaQuotes Ltd.



Vérification de l'échantillon :

```
if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alerte(" Vérifiez si le trading automatisé est autorisé dans les paramètres du t
else
  {
   if(!MQLInfoInteger(MQL_TRADE_ALLOWED))
      Alerte(" Le trading automatisé est interdit dans les paramètres du programme
  }
```

Vérifier si le trading est autorisé pour n'importe quel Expert Advisor/script sur le compte actuel

Le trading automatique peut être désactivé sur le serveur de trading. Vérification de l'échantillon :

```
if(!AccountInfoInteger(ACCOUNT_TRADE_EXPERT))
   Alerte(" Le trading automatisé est interdit pour le compte ",AccountInfoInteger
   " sur le serveur de trading");
```

Si le trading automatisé est désactivé sur le compte, les opérations de trading des Expert Advisors/scripts ne seront pas exécutées.

Vérification si le trading est autorisé sur le compte actuel

Dans certains cas, les opérations de trading sont désactivées pour un compte spécifique - ni le trading manuel, ni le trading automatisé ne peuvent être effectués. Vérification de l'échantillon lorsqu'un mot de passe investisseur a été utilisé pour se connecter à un compte de trading :

```
if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
   Commentaire("Le trading est interdit pour le compte ",AccountInfoInteger(ACCOUNT
         ".\n Il est possible que le mot de passe investisseur ait été utilisé pour
         "\n Vérifiez le journal du terminal pour l'entrée suivante :",
```

© 2000-2024, MetaQuotes Ltd.

```
            "\n\'",AccountInfoInteger(ACCOUNT_LOGIN),"\': le trading a été désactivé -
```

AccountInfoInteger(ACCOUNT_TRADE_ALLOWED) peut retourner **faux** dans les cas suivants :

- pas de connexion au serveur de trading. Cela peut être vérifié à l'aide de TerminalInfoInteger(TERMINAL_CONNECTED) ;
- compte de trading en mode lecture seule (archivé) ;
- le trading sur le compte est désactivé sur le serveur de trading ;
- la connexion à un compte de trading a été effectuée en mode Investisseur.

Voir également

Propriétés du Terminal Client, Propriétés du Compte, Propriétés du programme en cours d'éxécution sur MQL5

© 2000-2024, MetaQuotes Ltd.

Les événements du terminal de client

Init

Tout de suite après que le terminal de client chargera le programme (l'expert ou l'indicateur d'utilisateur) et lancera le procès de l'initialisation des variables globales, l'événement Init, sera envoyé, qui sera traité par la fonction OnInit(), si elle existe. Cet événement est généré aussi après le remplacement de l'instrument financier et/ou la période du graphique, après la recompilation du programme dans le rédacteur MetaEditor, après le changement des paramètres d'entrée de la fenêtre du réglage de l'expert ou l'indicateur d'utilisateur. Le conseiller est initialisé aussi après le changement du compte. Pour les scripts l'événement Init ne se généré pas.

Deinit

Devant la déinitialisation des variables globales et le déchargement du programme (l'expert ou l'indicateur d'utilisateur) le terminal de client envoie au programme l'événement Deinit. L'événement Deinit est généré aussi à l'achèvement du travail du terminal de client, à la clôture du graphique, directement devant le changement de l'instrument financier et/ou la période du graphique, à la recompilation réussie du programme, au changement des paramètres d'entrée, ainsi qu'au changement du compte.

La raison de deinitialisation peut être obtenue du paramètre, transmis à la fonction OnDeinit(). L'exécution de la fonction OnDeinit () se limite à 2.5 secondes. Si pour ce temps la fonction n'a pas fini le travail, son exécution s'achève forcément. Pour les scripts l'événement Deinit ne se généré pas.

Start

Start est un évènement spécial pour lancer un script ou un script après l'avoir chargé. Il est géré avec la fonction OnStart. L'évènement Start n'est pas passé aux EA et aux indicateurs personnalisés.

NewTick

L'événement NewTick est généré à l'entrée des nouvelles cotations et est traité par la fonction OnTick() chez les conseillers adjoints. Si à l'entrée de la nouvelle cotation on accomplissait la fonctio OnTick, lancée sur la cotation précédente, la cotation reçue sera ignorée par le conseiller, puisque l'événement correspondant n'est pas mis au tour des événements de l'expert.

Toutes les nouvelles cotations qui sont venues pendant l'exécution du programme sont ignorées par le programme jusqu'à ce que s'achève l'exécution suivante de la fonction OnTick(). Après cela la fonction sera lancée seulement après l'arrivée de la nouvelle cotation suivante. L'événement NewTick n'est pas généré à l'interdiction insérée de l'utilisation des conseillers (le bouton "Permettre/interdire les conseillers").

L'événement NewTick est généré indépendamment du fait, si le commerce automatique est permis ou interdit (le bouton "Permettre/interdire le commerce automatique"). L'interdiction du commerce automatique signifie seulement l'interdiction à l'expédition des requêtes commerciales de l'expert, le travail de l'expert ne cesse pas.

L'interdiction du commerce automatique par voie de la pression sur le bouton indiqué n'interrompt pas l'exécution de la fonction courrante OnTick().

Calculate

© 2000-2024, MetaQuotes Ltd.

L'événement Calculate est généré seulement pour les indicateurs après l'envoi de l'événement Init et à n'importe quel changement des données de prix. Il est traité par la fonction OnCalculate.

Timer

L'événement Timer est généré périodiquement par le terminal de client pour l'expert, qui a activé le minuteur à l'aide de la fonction EventSetTime. D'habitude cette fonction est appelé dans une fonction OnInit. Le traitement de l'événement Timer est produit par la fonction OnTimer. A l'achèvement du travail de l'expert il est nécessaire de supprimer le minuteur créé l'aide de EventKillTimer, qui est d'habitude appelé à la fonction OnDeinit.

Trade

L'événement Trade est généré à l'achèvement de l'opération commerciale sur le serveur commercial. Le traitement de l'événement Trade est produit par la fonction OnTrade() pour les opérations suivantes commerciales:

- L'installation, la modification ou l'effacement de l'ordre remis;
- l'annulation de l'ordre remis au manque des moyens ou à l'expiration;
- le fonctionnement de l'ordre remis;
- l'ouverture, le supplément ou la clôture de la position (ou de la partie de la position);
- la modification de la position ouverte (le changement des Stops).

TradeTransaction

A la suite de l'exécution des actions définies avec le compte commercial, son état change. Ces actions comprennent:

- L'envoi de la demande commerciale par n'importe quelle application MQL5- dans le terminal de client à l'aide des fonctions OrderSend et OrderSendAsync et son exécution ultérieure;
- L'envoi de la demande commerciale par l'interface graphique du terminal et son exécution ultérieure;
- le fonctionnement des ordres remis et des ordres stop sur le serveur;
- L'exécution des opérations sur le côté du serveur commercial.

A la suite de ces actions, les transactions commerciales sont exécutées pour le compte:

- le traitement de la demande commerciale;
- le changement des ordres ouverts;
- le changement de l'histoire des ordres;
- le changement de l'histoire des marchés;
- le changement des positions.

Par exemple, à l'envoi de l'ordre de marché sur l'achat, il est traité, l'ordre correspondant sur l'achat est créé pour le compte, se passe l'exécution de l'ordre, son suppression de la liste des ouverts, l'ajout à l'histoire des ordres, ensuite le marché correspondant est ajouté à l'histoire et une nouvelle position est créé. Toutes ces actions sont les transactions commerciales. L'arrivée de chaque telle transaction au terminal est l'événement TradeTransaction. Cet événement est traité par la fonctionOnTradeTransaction.

Tester

© 2000-2024, MetaQuotes Ltd.

L'événement Tester est généré à l'achèvement du test de l'expert des données historiques. Le traitement de l'événement Tester est produit par la fonction OnTester().

TesterInit

L'événement TesterInit est généré au lancement de l'optimisation dans le testeur des stratégies avant le premier passage. Le traitement de l'événement TesterInit est produit par la fonctionOnTesterInit().

TesterPass

L'événementTesterPass est généré à l'entréed'une nouvelle trame des données. Le traitement de l'événement TesterPass est produit par la fonction OnTesterPass().

TesterDeinit

L'événement TesterDeinit est généré à l'issue de l'optimisation de l'expert dans le testeur des stratégies. Le traitement de l'événement TesterDeinit est produit par la fonctionOnTesterDeinit().

ChartEvent

L'événement ChartEvent est généré par le terminal de client pendant le travail d'utilisateur avec le graphique:

- la pression du clavier, quand la fenêtre du graphique se trouve dans le tour;
- la création de l'objet graphique;
- l'effacement l'objet graphique;
- le clic de la souris sur l'objet graphique, appartenant au graphique;
- le déplacement de l'objet graphique à l'aide de la souris;
- la fin de l'édition du texte dans le champ d'entrée de l'objet graphique LabelEdit.

Il y a aussi un événement d'utilisateur ChartEvent, qui peut être envoyé à l'expert par chaque programme mql5 à l'aide de la fonction EventChartCustom. L'événement est traité par la fonction OnChartEvent.

BookEvent

L'événement BookEvent est généré par le terminal de client au changement de l'état du profondeur de marché et est traité par la fonction OnBookEvent. Pour que le terminal de client commence à générer les événements BookEvent selon le symbole concret, c'est assez de s'inscrire préalablement sur la réception de ces événements pour ce symbole à l'aide de la fonction MarketBookAdd.

Pour désabonner de la réception de l'événement BookEvent selon le symbole, il est nécessaire d'appeler la fonction MarketBookRelease. L'événement BookEvent est celui de diffusion – cela signifie qu'il est suffisant à un expert de souscrire à la réception de l'événement BookEvent à l'aide de la fonction MarketBookAdd, et tous les autres experts qui ont le gestionnaire OnBookEvent, recevrons cet événement. C'est pourquoi il est nécessaire d'analyser le nom du symbole, qui est transmis au gestionnaire à titre du paramètre.

Voir aussi

Les fonctions du traitement des événements, L'exécution des programmes

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Ressources

Utilisation du graphique et du son dans les programmes MQL5

Les programmes MQL5 permettent de travailler avec des fichiers sonores et graphiques :

- PlaySound() joue un fichier audio ;
- ObjectCreate() permet de créer des interfaces utilisateur à l'aide des objets graphiques OBJ_BITMAP et OBJ_BITMAP_LABEL.

PlaySound()

Exemple d'appel de la fonction PlaySound ():

```
//+------------------------------------------------------------------+
//| La fonction appelle OrderSend() et joue le son                   |
//+------------------------------------------------------------------+
void OrderSendWithAudio(MqlTradeRequest  &request, MqlTradeResult &result)
  {
 //--- envoie une demande au serveur
   OrderSend(request,result);
 //--- si la demande est acceptée, joue un son Ok.wav
   if(result.retcode==TRADE_RETCODE_PLACED) PlaySound("Ok.wav");
   //--- si elle échoue, joue une alarme à partir du fichier timeout.wav
   else PlaySound("timeout.wav");
  }
```

Cet exemple montre comment jouer des sons à partir des fichiers Ok.wav et timeoit.wav, qui sont inclus dans le package du terminal standard. Ces fichiers se trouvent dans le dossier **terminal_directory\Sounds**. Ici **terminal_directory** est le dossier à partir duquel le terminal Client MetaTrader 5 est démarré. L'emplacement du répertoire du terminal peut être trouvé à partir d'un programme MQL5 de la manière suivante :

```
//--- Dossier dans lequel les données du terminal sont stockées
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

Vous pouvez utiliser des fichiers audio non seulement à partir du dossier **terminal_directory\Sounds** mais aussi de n'importe quel sous-dossier qui se trouve dans le dossier **terminal_data_directory\MQL5.** Vous pouvez trouver l'emplacement du répertoire des données du terminal à partir du menu du terminal "Fichier" - "Ouvrir le répertoire des données" ou en utilisant la méthode du programme :

```
//--- Dossier dans lequel les données du terminal sont stockées
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

Par exemple, si le fichier audio Demo.wav se trouve dans le terminal_data_directory\MQL5\Files, l'appel à la fonction PlaySound() doit être écrit ainsi :

```
//--- joue un fichier audio Demo.wav à partir du terminal_directory_data\MQL5\Files
   PlaySound("\\Files\\Demo.wav");
```

© 2000-2024, MetaQuotes Ltd.

Veuillez noter que le chemin vers le fichier est écrit en utilisant le symbole "\" dans le commentaire, et dans la fonction, le double backslash "\\" est utilisé.

Lorsque vous spécifiez le chemin d'accès, utilisez toujours le double backslash comme séparateur, car un seule backslash est le symbole de commande pour le compilateur lorsqu'il traite des chaînes constantes et des constantes de caractères dans le code source du programme.

Appelez la fonction PlaySound() avec le paramètre NULL pour arrêter la lecture :

```
//--- Appelez la fonction PlaySound () avec le paramètre NULL pour arrêter la lecture
   PlaySound(NULL);
```

ObjectCreate()

Exemple d'un Expert Advisor qui crée une "Etiquette graphique" (OBJ_BITMAP_LABEL) en utilisant la fonction ObjectCreate().

```
string label_name="currency_label";        // le nom de l'objet OBJ_BITMAP_LABEL
string euro      ="\\Images\\euro.bmp";    // le chemin vers le fichier terminal_data_
string dollar    ="\\Images\\dollar.bmp";  // le chemin vers le fichier terminal_data_
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- créer un bouton OBJ_BITMAP_LABEL, s'il n'a pas été encore crée
   if(ObjectFind(0,label_name)<0)
     {
      //--- essaye de créer l'objet OBJ_BITMAP_LABEL
      bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
      if(created)
        {
         //--- place le bouton dans le coin supérieur gauche du graphique
         ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
 //--- configuration des propriétés d'un objet
         ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
         ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
         //--- réinitialisation du code de la dernière erreur à 0
         ResetLastError();
         //--- chargement d'une image pour indiquer le bouton en mode "appuyé"
         bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
         //--- vérification du résultat
         if(!set)
           {
            PrintFormat("Échec de chargement de l'image à partir du fichier %s. Le coc
           }
         ResetLastError();
         //--- chargement d'une image pour indiquer le bouton en mode "non appuyé"
```

© 2000-2024, MetaQuotes Ltd.

```
         set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);

         if(!set)
           {
            PrintFormat("Échec de chargement de l'image à partir du fichier %s. Le coc
           }
         //--- envoi d'une commande pour la mise à jour du graphique de sorte que le b
         ChartRedraw(0);
        }
      else
        {
         //--- impossible de créer l'objet
         PrintFormat("Impossible de créer l'objet OBJ_BITMAP_LABEL. Le code de l'erreu
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinstallation de l'Expert                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- suppression d'un objet du graphique
   ObjectDelete(0,label_name);
  }
```

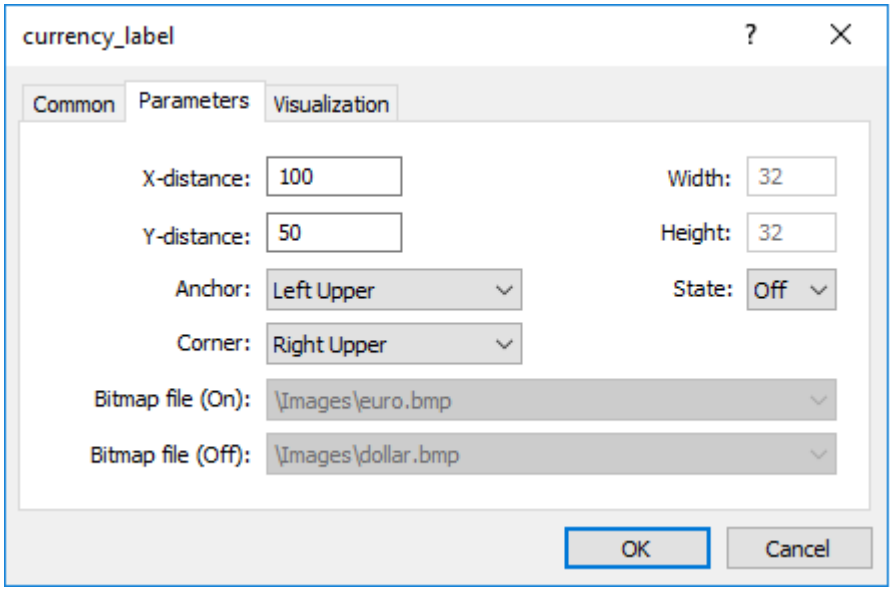
La création et la configuration d'un objet graphique nommé currency_label sont effectuées dans la fonction OnInit(). Les chemins d'accès aux fichiers graphiques sont définis dans les variables globales **euro** et **dollar**, un double backslash est utilisé comme séparateur :

```
string euro      ="\\Images\\euro.bmp";     // le chemin vers le fichier terminal_data_
string dollar    ="\\Images\\dollar.bmp";   // le chemin vers le fichier terminal_data_
```

Les fichiers se trouvent dans le dossier **terminal_data_directory**\MQL5\Images.

L'objet OBJ_BITMAP_LABEL est un bouton, qui affiche l'une des deux images en fonction de l'état du bouton (appuyé ou non) : euro.bmp ou dollar.bmp.

© 2000-2024, MetaQuotes Ltd.



La taille du bouton avec une interface graphique est automatiquement ajustée à la taille de l'image. L'image est modifiée avec le bouton gauche de la souris sur l'objet OBJ_BITMAP_LABEL (**l'option "Désactiver la sélection" doit être cochée dans les propriétés**). L'objet OBJ_BITMAP est créé de la même manière et est destiné à la création de l'arrière-plan avec le dessin demandé.

La valeur de la propriété OBJPROP_BMPFILE, qui est responsable de l'apparition des objets OBJ_BITMAP et OBJ_BITMAP_LABEL, peut être modifié dynamiquement. Cela permet de créer différentes interfaces utilisateur interactives pour les programmes MQL5.

Inclusion de ressources vers des fichiers exécutables lors de la compilation des programmes MQL5

Un programme MQL5 peut avoir besoin de beaucoup de ressources différentes téléchargeables sous forme de fichiers image et son. Afin d'éliminer la nécessité de transférer tous ces fichiers lors du déplacement d'un fichier exécutable dans MQL5, il faut utiliser la directive du compilateur #resource:

```
#resource doit être utilisé : #resource path_to_resource_file
```

La commande #resource indique au compilateur que la ressource du chemin spécifié **path_to_resource_file** doit être inclus dans le fichier exécutable EX5. Ainsi, toutes les images et tous les sons nécessaires peuvent être placés directement dans un fichier EX5, de sorte qu'il n'y a pas besoin de transférer séparément les fichiers utilisés, si vous voulez exécuter le programme sur un autre terminal. N'importe quel fichier EX5 peut contenir des ressources et n'importe quel programme EX5 peut utiliser les ressources d'un autre programme EX5.

Les fichiers en format BMP et WAV sont automatiquement compressées avant de les inclure dans un fichier EX5. Cela signifie que, en plus de la création de programmes complets dans MQL5, utiliser les ressources permet également de réduire la taille totale des fichiers nécessaires lors de l'utilisation des graphiques et des sons, par rapport à la façon traditionnelle d'écriture des programmes MQL5 .

La taille du fichier de ressources ne doit pas dépasser 16 Mo.

© 2000-2024, MetaQuotes Ltd.

Recherche des ressources spécifiées par le compilateur

La ressource est insérée en utilisant la commande #resource "<chemin vers le fichier ressource>"

```
#resource "<chemin vers le fichier ressource>"
```

La longueur de la chaîne constante <chemin vers le fichier ressource> ne doit pas dépasser 63 caractères.

Le compilateur recherche la ressource selon le chemin spécifié dans l'ordre suivant :

- si le séparateur backslash "\" (écrit comme "\\") est placé au début du chemin, il recherche la ressource par rapport au répertoire **terminal_data_directory** \MQL5\,
- s'il n'y a pas de backslash, il recherche la ressource par rapport à l'emplacement du fichier source, dans lequel la ressource est écrite.

Le chemin de la ressource ne peut pas contenir les sous-chaînes "..\\" et ":\\"

Exemples d'insertion de ressources :

```
//--- indication des ressources
#resource "\\Images\\euro.bmp" // euro.bmp se trouve dans terminal_data_directory\MQL5
#resource "picture.bmp"        // picture.bmp se trouve dans le même répertoire que le
#resource "Resource\\map.bmp"  // la ressource se trouve dans le dossier source_file_c

//--- indication incorrecte des ressources
#resource ": picture_2.bmp"     // ne doit pas contenir ":"
#resource "..\\picture_3.bmp"  // ne doit pas contenir ".."
#resource "\\Files\\Images\\Folder_First\\My_panel\\Labels\\too_long_path.bmp" // plus
```

Utilisation des ressources

Nom de la ressource

Quand la ressource est déclarée en utilisant la directive #resource, on peut l'utiliser dans n'importe quelle partie du programme. Le nom de la ressource devient son chemin sans le backslash au début de la ligne qui spécifie le chemin d'accès à la ressource. Pour utiliser votre propre ressource dans le code, il faut ajouter un signe spécial "::" avant le nom de la ressource.

Exemples :

```
//--- exemples de spécification des ressources et de leurs noms dans les commentaires
#resource "\\Images\\euro.bmp"          // le nom de la ressource - Images\euro.bmp
#resource "picture.bmp"                 // le nom de la ressource - picture.bmp
#resource "Resource\\map.bmp"           // le nom de la ressource - Resource\map.bmp
#resource "\\Files\\Pictures\\good.bmp" // le nom de la ressource - Files\Pictures\goo
#resource "\\Files\\Demo.wav";          // le nom de la ressource - Files\Demo.wav"
#resource "\\Sounds\\thrill.wav";       // le nom de la ressource - Sounds\thrill.wav"
...

//--- utilisation des ressources
```

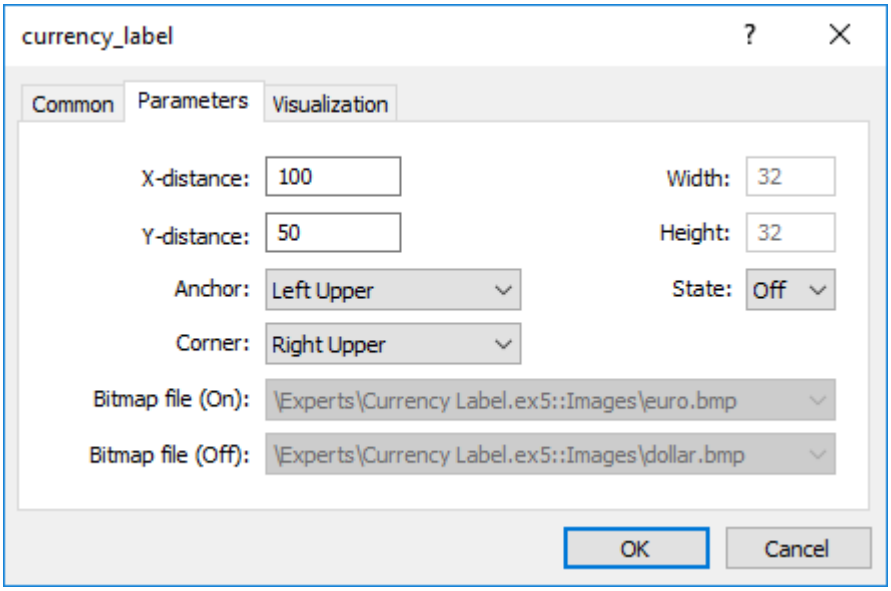
© 2000-2024, MetaQuotes Ltd.

```
ObjectSetString(0,bitmap_name,OBJPROP_BMPFILE,0,"::Images\\euro.bmp");
...
ObjectSetString(0,my_bitmap,OBJPROP_BMPFILE,0,"::picture.bmp");
...
set=ObjectSetString(0,bitmap_label,OBJPROP_BMPFILE,1,"::Files\\Pictures\\good.bmp");
...
PlaySound("::Files\\Demo.wav");
...
PlaySound("::Sounds\\thrill.wav");
```

Il convient de noter que pendant l'installation de l'image de la ressource pour les objets OBJ_BITMAP et OBJ_BITMAP_LABEL, on ne peut pas changer manuellement la valeur de la propriété OBJPROP_BMPFILE. Par exemple, supposons que nous utilisons les ressources des fichiers euro.bmp et dollar.bmp. pour créer OBJ_BITMAP_LABEL.

```
#resource "\\Images\\euro.bmp";    // euro.bmp est situé dans le terminal_data_directo
#resource "\\Images\\dollar.bmp";  // dollar.bmp est situé dans le terminal_data_direc
```

Lorsque vous affichez les propriétés de cet objet, nous voyons que les propriétés BitMap File (On) et BitMap File (Off) sont grisés et ne peuvent pas être changées manuellement :



Utiliser les ressources des autres programmes mql5

L'utilisation des ressources a un autre avantage - on peut utiliser les ressources de chaque fichier EX5 dans chaque programme mql5. Ainsi, on peut utiliser les ressources d'un fichier EX5 dans les autres programmes mql5.

Pour utiliser le nom de la ressource d'un fichier tiers, il faut le spécifier comme <chemin_du nom_du fichier_EX5>::<nom_de la ressource>. Par exemple, si la ressource pour une image dans le fichier triangle.bmp est spécifiée dans le script Draw_Triangles_Script.mq5 :

```
#resource "\\Files\\triangle.bmp"
```

© 2000-2024, MetaQuotes Ltd.

Puis son nom pour l'utilisation dans le script lui-même comme "Files\triangle.bmp", et pour l'utiliser, il faut ajouter un signe spécial "::" pour le nom de la ressource.

```
//--- utilisation de la ressource dans le script
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

Pour utiliser la même ressource d'un autre programme, par exemple à partir d'un EA, il faut ajouter au nom de la ressource le chemin vers le fichier EX5 par rapport au dossier **terminal_data_directory\MQL5** et le nom du fichier EX5 du script - **Draw_Triangles_Script.ex5**. Si le script se trouve dans le dossier standard **terminal_data_directory\MQL5**Scripts\, il faut écrire l'appel de cette façon :

```
//--- utilisation de la ressource à partir d'un script dans un EA
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\\Draw_Triangles_Script.e
```

Si le chemin d'accès au fichier exécutable n'est pas spécifié lors de l'appel de la ressource d'un autre EX5, le fichier exécutable est recherché dans le même dossier qui contient le programme qui appelle la ressource. Cela signifie que si un EA appelle une ressource à partir du fichier Draw_Triangles_Script.ex5 sans indication du chemin, comme ceci :

```
//--- appel d'une ressource du script dans un EA sans spécifier le chemin
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files\\
```

le fichier sera recherché dans le dossier **terminal_data_directory\MQL5**Experts\, si l'EA se trouve dans le dossier **terminal_data_directory\MQL5**Experts\.

Travailler avec les indicateurs personnalisés inclus en tant que ressources

Un ou plusieurs indicateurs personnalisés peuvent être nécessaires pour le fonctionnement des applications MQL5. Chacun d'entre eux peuvent être inclus dans le code d'un programme MQL5 exécutable. L'insertion des indicateurs comme des ressources simplifie la distribution des applications.

Voici un exemple d'insertion et d'utilisation d'un indicateur personnalisé SampleIndicator.ex5 situé dans le dossier : **terminal_data_folder** \\MQL5\Indicators\ :

```
//+------------------------------------------------------------------+
//|                                                     SampleEA.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#resource "\\Indicators\\SampleIndicator.ex5"
int handle_ind;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   handle_ind=iCustom(_Symbol,_Period,"::Indicators\\SampleIndicator.ex5");
   if(handle_ind==INVALID_HANDLE)
```

```
      {
       Print("Expert : appel à iCustom : Code d'erreur=",GetLastError());
       return(INIT_FAILED);
      }
//--- ...
   return(INIT_SUCCEEDED);
  }
```

Le cas où un indicateur personnalisé dans la fonction OnInit() crée une ou plusieurs copies de lui-même, exige une attention particulière. Rappellons que la ressource doit être spécifiée de la manière suivante : **<path_EX5_file_name>::<resource_name>**.

Par exemple, si l'indicateur SampleIndicator.ex5 fait partie de l'EA SampleEA.ex5 comme une ressource, le chemin vers lui-même, spécifié lors de l'appel iCustom() dans la fonction d'initialisation de l'indicateur personnalisé sera ainsi : "\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5". Lorsque ce chemin est défini explicitement, l'indicateur personnalisé SampleIndicator.ex5 sera attaché à l'EA SampleEA.ex5 et perdra la capacité de travailler de façon indépendante.

Le chemin d'accès à lui-même peut être reçu en utilisant la fonction GetRelativeProgramPath(), dont l'exemple de l'utilisation est fourni ci-dessous :

```
//+------------------------------------------------------------------+
//|                                              SampleIndicator.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property indicator_separate_window
#property indicator_plots 0
int handle;
//+------------------------------------------------------------------+
//| Fonction d'initialisation d'un indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mauvaise façon de fournir un lien a lui-même
//--- string path="\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5";
//--- bonne façon de recevoir un lien à lui-même
  string path=GetRelativeProgramPath();
//--- mapping des buffers de l'indicateur
   handle=iCustom(_Symbol,_Period,path,0,0);
   if(handle==INVALID_HANDLE)
     {
      Print("Indicateur : appel à iCustom : Code d'erreur=",GetLastError());
      return(INIT_FAILED);
     }
   else Print("Handle de l'indicateur=",handle);
//---
   return(INIT_SUCCEEDED);
  }
//+....
```

```
//+------------------------------------------------------------------+
//| GetRelativeProgramPath                                           |
//+------------------------------------------------------------------+
string GetRelativeProgramPath()
  {
   int pos2;
//--- obtient le chemin absolu vers le programme
   string path=MQLInfoString(MQL_PROGRAM_PATH);
//--- trouve la position de la sous-chaîne "\MQL5\"
   int    pos =StringFind(path,"\\MQL5\\");
//--- la sous-chaîne n'est pas trouvée - erreur
   if(pos<0)
      return(NULL);
//--- saut du répertoire "\MQL5"
   pos+=5;
//--- Sauts supplémentaires '\' symboles
   tandis que (StringGetCharacter(path,pos+1)=='\\')
      pos++;
//--- si c'est la ressource, retourne le chemin relatif au répertoire MQL5
   if(StringFind(path,"::",pos)>=0)
      return(StringSubstr(path,pos));
//--- trouve un séparateur pour le premier sous-répertoire MQL5 (par exemple, MQL5\Inc
//--- s'il est absent, retourne le chemin relatif au répertoire MQL5
   if((pos2=StringFind(path,"\\",pos+1))<0)
      return(StringSubstr(path,pos));
//--- retourne le chemin relatif au répertoire (par exemple, MQL5\Indicators)
   return(StringSubstr(path,pos2+1));
  }
//+------------------------------------------------------------------+
//| Fonction d'itération d'un indicateur personnalisé |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double& price[])
  {
//--- valeur de retour de prev_calculated pour le prochain appel
   return(rates_total);
  }
```

Resource variables

Les ressources peuvent être déclarées en utilisant des variables de ressources et traitées comme si elles étaient de variables du type correspondant. Format de déclaration :

```
#resource chemin_vers_le_fichier_de_ressource as type_de_la_variable_de_ressource nom_
```

© 2000-2024, MetaQuotes Ltd.

Exemples de déclarations :

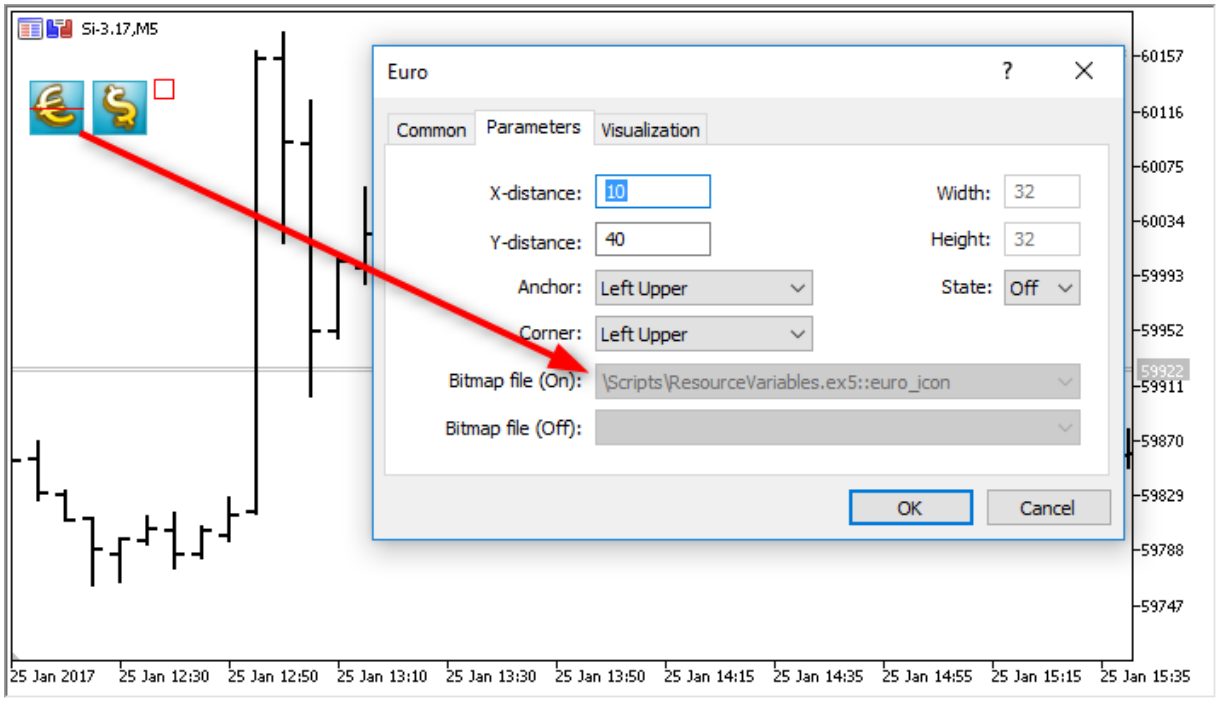
```
#resource "data.bin" as int ExtData[]              // déclare un tableau numérique cont
#resource "data.bin" as MqlRates ExtData[]         // déclare un tableau de structures
//--- chaînes de caractères
#resource "data.txt" as string ExtCode             // déclare une chaîne de caractères
//--- ressources graphiques
#resource "image.bmp" as bitmap ExtBitmap[]        // déclare un tableau à une dimensio
#resource "image.bmp" as bitmap ExtBitmap2[][]     // déclare un tableau à deux dimensi
```

Dans le cas d'une telle déclaration, les données de la ressources ne sont utilisables que via la variable, l'auto-adressage via "::<nom de la ressource>" ne fonctionne pas.

```
#resource "\\Images\\euro.bmp" as bitmap euro[][]
#resource "\\Images\\dollar.bmp"
//+------------------------------------------------------------------+
//|  Fonction de création de l'OBJ_BITMAP_LABEL avec une ressource |
//+------------------------------------------------------------------+
void Image(string name,string rc,int x,int y)
  {
   ObjectCreate(0,name,OBJ_BITMAP_LABEL,0,0,0);
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
   ObjectSetString(0,name,OBJPROP_BMPFILE,rc);
  }
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- affiche la taille de l'image [largeur, hauteur] stockée dans la variable de ress
   Print(ArrayRange(euro,1),", ",ArrayRange(euro,0));
//--- change l'image de euro - dessine des bandes horizontales rouges au milieu
   for(int x=0;x<ArrayRange(euro,1);x++)
      euro[ArrayRange(euro,1)/2][x]=0xFFFF0000;
//--- crée la ressource graphique en utilisant la varaiable de ressource
   ResourceCreate("euro_icon",euro,ArrayRange(euro,1),ArrayRange(euro,0),0,0,ArrayRang
//--- crée l'objet étiquette graphique Euro, sur laquelle l'image de la ressource euro
   Image("Euro","::euro_icon",10,40);
//--- autre méthode appliquant la ressource, il n'est pas possible de dessiner dessus
   Image("USD","::Images\\dollar.bmp",15+ArrayRange(euro,1),40);
//--- la méthode directe d'adressage de la ressource euro.bmp n'est pas disponible pui
   Image("E2","::Images\\euro.bmp",20+ArrayRange(euro,1)*2,40); // execution time erro
  }
```

Résultat d'exécution du script - seuls deux objets OBJ_BITMAP_LABEL sur trois sont créés. L'image du premier objet a les bandes rouges au milieu.

© 2000-2024, MetaQuotes Ltd.



Un avantage important d'appliquer les ressources est que les fichiers de ressources sont automatiquements compressés avant d'être inclus dans le fichier exécutable EX5 avant sa compilation. Ainsi, l'utilisation des variables de ressources vous permet de mettre toutes les données nécessaires dans le fichier exécutable EX5 mais aussi de réduire le nombre et la taille totale des fichiers comparé à la méthode habituelle d'écriture des programmes MQL5.

L'utilisation des variables de ressources est particulièrement pratique pour publier les produits sur le Market.

Fonctionnalités

- Le type spécial de variable de ressource *bitmap* informe le compilateur que la ressource est une image. Ce type de variables est de type uint.
- La variable de ressource tableau de type *bitmap* peut avoir 2 dimensions. Dans ce cas, la taille du tableau est définie comme [hauteur_image][largeur_image]. Si un tableau à 1 dimension est spécifié, le nombre d'éléments est égal à hauteur_image * largeur_image.
- Lors du chargement d'une image 24 bits, le composant canal alpha est défini à 255 pour tous les pixels de l'image.
- Lors du chargement d'une image 32 bits sans canal alpha, le composant canal alpha est également défini à 255 pour tous les pixels de l'image.
- Lors du chargement d'une image 32 bits avec un canal alpha, les pixels ne sont pas traités.
- La taille du fichier de ressource ne peut pas être supérieure à 128 Mo.
- La détection automatique de l'encodage par la présence du BOM (en-tête) est effectuée pour les fichiers de chaînes de caractères. Si le BOM est absent, l'encodage est défini par le contenu du fichier. Le fichiers avec un encodage ANSI, UTF-8 et UTF-16 sont supportés. Toutes les chaînes de caractères sont converties en Unicode lors de la lecture des données des fichiers.

Programmes OpenCL

© 2000-2024, MetaQuotes Ltd.

Utiliser les variables de ressource de chaînes de caractères facilite grandement le développement de certains programmes. Par exemple, vous pouvez écrire le code d'un programme OpenCL dans un fichier CL séparé et l'inclure ensuite comme une simple chaîne de caractères dans les ressources de votre programme MQL5.

```
#resource "seascape.cl" as string cl_program
...
int context;
if((cl_program=CLProgramCreate(context,cl_program)!=INVALID_HANDLE)
  {
   //--- effectue d'autres actions avec le programme OpenCL
  }
```

Dans cet exemple, vous auriez à écrire le code entier comme une seule grosse chaîne de caractères si la variable de ressource *cl_program* n'avait pas été utilisée.

Voir aussi

ResourceCreate(), ResourceSave(), PlaySound(), ObjectSetInteger(), ChartApplyTemplate(), Fonctions de fichier

© 2000-2024, MetaQuotes Ltd.

L'appel des fonctions importées

Pour importer des fonctions pendant l'exécution du programme mql5 on utilise un enchaînement . Cela signifie que si dans le programme il y a un appel de la fonction importée, le module correspondant (ex5 ou dll) est chargé en train du chargement du programme. Les bibliothèques MQL5 et DLL sont accomplies dans le thread du module appelant.

Il n'est pas recommandé d'utiliser le nom entièrement qualifié du module chargé comme *Drive:\Directory\FileName.Ext*. Les bibliothèques MQL5 sont chargées du classeur *terminal_dir\MQL5\Libraries*. Si la bibliothèque n'était pas trouvée, on produit la tentative de charger la bibliothèque du classeur *terminal_dir\experts*.

Les bibliothèques systémiques (DLL) sont chargées selon les règles du système opérationnel. Si la bibliothèque est déjà chargée (par exemple, d'un autre expert et même d'un autre terminal de client lancé parallèlement), l'appel s'adresse à la bibliothèque déjà chargée. Dans le cas contraire la recherche va dans l'héritage suivant:

1. Le répertoire d'où on lançait le module, important dll. Sous le module on considère l'expert, le script, l'indicateur ou la bibliothèque EX5;
2. Le répertoire le répertoire_du_terminal_données\MQL5\Libraries (TERMINAL_DATA_PATH\MQL5\Libraries);
3. Le répertoire, d'où on lançait le terminal de client MetaTrader 5;
4. Le répertoire systémique;
5. Le répertoire Windows;
6. Le répertoire courant;
7. Les répertoires, énumérés dans une variable de système de l'entourage PATH.

Si la bibliothèque DLL utilise dans le travail l'autre DLL, en cas de l'absence de la deuxième DLL, la première ne pourra pas être chargé.

Avant le chargement de l'expert (le script, l'indicateur) on forme le répertoire total de toutes les modules de bibliothèque EX5, qu'on suppose d'utiliser comme de l'expert récupéré (le scénario, l'indicateur), et des bibliothèques de cette liste. On assure ainsi le chargement unique des modules plusieurs fois utilisées de bibliothèque EX5. Les bibliothèques se servent des variables prédéterminées de l'expert qui les a provoqué (le script, l'indicateur).

La recherche de la bibliothèque importée EX5 est produite dans la séquence suivante:

1. Le répertoire, le chemin vers lequel est donné par rapport au répertoire important EX5 de l'expert (le script, l'indicateur);
2. Le répertoire le répertoire_du terminal\MQL5\Libraries;
3. Le répertoire MQL5\Libraries dans le répertoire commun de tous les terminaux de client MetaTrader 5(Common\MQL5\Libraries).

Les fonctions importées de DLL au programme mql5, doivent assurer l'accord sur les liens, accepté pour les fonctions Windows API. Pour la garantie d'un tel accord dans le texte initial des programmes écrits dans les langages C ou C ++ on utilise le mot-clé __stdcall, qui est spécifique pour les compilateurs de la compagnie Microsoft(r). L'accord est caractérisé par:

- la fonction appelante (dans notre cas c'est le programme mq5) doit "voir" le prototype de la fonction appelée (importée de DLL), pour mettre correctement les paramètres à la pile;

© 2000-2024, MetaQuotes Ltd.

- la fonction appelante (dans notre cas c'est le programme mq5) met les paramètres sur la pile dans l'ordre inverse, de droite à gauche - notamment en tel ordre la fonction importée lit les paramètres transmis à elle;
- les paramètres sont transmis selon la valeur, à l'exception de ceux qui sont transmis évidemment selon le lien (dans notre cas des lignes)
- la fonction importée, en lisant les paramètres transmis à elle, nettoie la pile.

A la description du prototype de la fonction importée on peut utiliser les paramètres avec les valeurs par défaut.

En cas si la bibliothèque correspondante n'a pas pu être chargée, ou on établit l'interdiction à l'utilisation DLL, ou la fonction importée n'était pas trouvée – l'expert arrête le travail avec le message correspondant "expert stopped" au journal. Ainsi l'expert ne fonctionnera pas avant que on ne le réinitialise pas. L'expert peut être réinitialisé à la suite de la recompilation ou après l'ouverture du tableau des propriétés de l'expert et la pression du bouton OK.

La transmission des paramètres.

Tous les paramètres des types simples sont transmis selon la valeur, si n'est pas indiqué évidemment qu'ils sont transmis selon le lien. A la transmission de la ligne on transmet l'adresse du tampon de la ligne copiée; si la ligne est transmise selon le lien, à la fonction importée de DLL, on transmet l'adresse du tampon notamment de cette ligne sans copiage.

Les structures, contenant les tableaux dynamiques, les lignes, les classes, d'autres structures complexes, ainsi que les tableaux dynamiques ou statiques des objets énumérés, ne peuvent pas être transmises à titre du paramètre à la fonction importée.

A la transmission à DLL du tableau toujours (indépendamment du drapeau AS_SERIES) on transmet l'adresse du début du tampon des données. La fonction au-dedans de DLL ne sait rien du drapeau AS_SERIES, le tableau transmis est le tableau statique de la longueur inconnue, pour l'indication de la grandeur du tableau il faut utiliser une paramétrisation supplémentaire.

© 2000-2024, MetaQuotes Ltd.

Les erreurs de l'exécution

Dans le sous-système exécutif du terminal de client il y a une possibilité de sauvegarde du code de l'erreur dans le cas de son occurrence à l'exécution du programme mql5. Pour chaque programme mql5 exécuté on prévoit une variable prédéterminée _LastError.

Avant le lancement la fonction OnInit la variable _LastError remet au zéro. A l'apparition de la situation fausse pendant les calculs ou en train de l'appel de la fonction insérée la variable _LastError prend le code correspondant de l'erreur. On peut recevoir la valeur sauvegerdé dans cette variable à l'aide de la fonction GetLastError().

Il y a une série d'erreurs critiques, à l'apparition de lesquelles l'exécution du programme s'interrompt immédiatement:

- la division en le zéro;
- la sortie en dehors du tableau;
- l'utilisation de l'indicateur sur l'objet incorrect;

© 2000-2024, MetaQuotes Ltd.

CopySpreadTest de Stratégies de Trading

L'idée du trading automatisé est séduisante par le fait que le robot de trading peut fonctionner non-stop 24 heures sur 24, 7 jours sur 7. Le robot ne connait pas la fatigue, le doute, la peur ; il est totalement exempt de tout problème psychologique. Il est nécessaire de formaliser clairement les règles de trading et de les mettre en œuvre dans les algorithmes ; le robot est alors prêt à travailler sans relâche. Mais d'abord, vous devez vous assurer que les deux conditions suivantes sont remplies :

- L'Expert Advisor effectue des opérations de trading en conformité avec les règles du système de trading ;
- La stratégie de trading mise en œuvre dans l'EA montre un profit dans l'historique.

Pour obtenir des réponses à ces questions, nous nous tournons vers le Strategy Tester, inclus dans le terminal client MetaTrader 5.

Cette section traite des caractéristiques de tests et d'optimisation du programme dans le Strategy Tester :

- Limitations des fonctions dans le Strategy Tester
- Les modes de génération des ticks
- Simulation du spread
- Utiliser des ticks réels pendant un test
- Les Variables globales du terminal client
- Calcul des indicateurs au cours des tests
- Chargement de l'historique au cours des tests
- Tests Multi-Devises
- Simulation du Temps dans le Strategy Tester
- Objets Graphiques pendant le Test
- La Fonction OnTimer() dans le Strategy Tester
- Fonction Sleep() dans le Strategy Tester
- Utilisation du Strategy Tester pour l'Optimisation des Problèmes dans les Calculs Mathématiques
- Synchronisation des Barres en mode "Prix d'Ouverture Uniquement"
- La fonction IndicatorRelease() dans le Strategy Tester
- Gestion des Evénements dans le Strategy Tester
- Agents de Test
- L'Echange des Données entre le Terminal et l'Agent
- Utilisation du Dossier Partagés de Tous les Terminaux Client
- Utilisation de DLL

Limites de mémoire et d'espace disque dans le MQL5 Cloud Network

© 2000-2024, MetaQuotes Ltd.

La limitation suivante s'applique aux optimisations exécutées dans le MQL5 Cloud Network : l'Expert Advisor ne doit pas écrire sur le disque plus de 4 Go d'informations ou utiliser plus de 4 Go de RAM . Si la limite est dépassée, l'agent du réseau ne pourra pas effectuer le calcul correctement et vous ne recevrez pas le résultat. Vous serez cependant facturé pour tout le temps passé sur les calculs.

Si vous avez besoin d'obtenir des informations à chaque passe d'optimisation, envoyez des images sans écrire sur le disque. Pour éviter d'utiliser les opérations sur les fichiers dans les Expert Advisors lors des calculs dans le MQL5 Cloud Network, vous pouvez utiliser la vérification suivante :

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

Limitations des Fonctions dans le Strategy Tester

Il y a des limites de fonctionnement pour certaines fonctions dans le Strategy Tester du terminal client.

Les Fonctions Comment(), Print() et PrintFormat()

Pour augmenter les performances, les fonctions Comment(), Print() et PrintFormat() ne sont pas exécutées lors de l'optimisation des paramètres du robot de trading. Une exception est l'utilisation de ces fonctions à l'intérieur du gestionnaire OnInit(). Cela vous permet de trouver facilement la cause des erreurs quand elles se produisent.

Les fonctions Alert(), MessageBox(), PlaySound(), SendFTP, SendMail(), SendNotification(), WebRequest()

Les fonctions Alert(), MessageBox(), PlaySound(), SendFTP(), SendMail(), SendNotification() et WebRequest() conçus pour interagir avec "le monde extérieur" ne sont pas exécutées dans le Strategy Tester.

Les modes de génération des ticks

Un Expert Advisor est un programme écrit en MQL5 qui est exécuté à chaque fois en réponse à un évènement extérieur. L'EA a une fonction correspondant à chaque événement prédéfini - ce sont les gestionnaires d'évènements.

© 2000-2024, MetaQuotes Ltd.

L'événement NewTick (changement de prix) est l'événement principal de l'EA et, par conséquent, nous avons besoin de générer une séquence de ticks pour tester l'EA. 3 modes de génération de ticks existent dans le Strategy Tester du terminal client MetaTrader 5 :

- Tous les ticks
- 1 Minute OHLC (les prix OKLC en barre 1 minute)
- Prix d'ouverture uniquement

Le mode de base de génération et le plus détaillé est le mode "Tous les ticks", les deux autres modes sont une simplification du mode de base et seront comparés avec le mode "Tous les ticks". Considérons les trois modes afin de comprendre leurs différences.

Tous les ticks

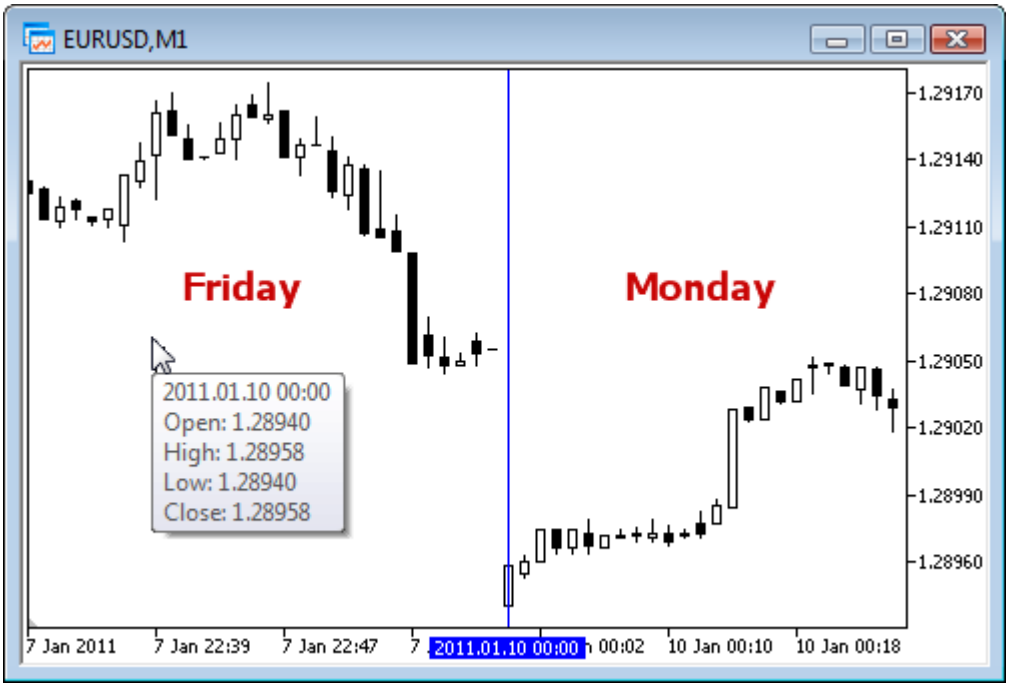
L'historique des cotations des instruments financiers est transmis du serveur de trading au terminal client MetaTrader 5 sous forme de barres 1 minute. Des informations détaillées sur l'apparition de demandes et sur la construction des cadres de temps requis sont disponibles dans le chapitre Organisation de l'Accès aux Données de la référence MQL5.

L'élément minimal de l'historique des prix est la barre en minute, à partir de laquelle vous pouvez obtenir des informations sur les quatre valeurs du prix :

- Open - le prix d'ouverture de la barre 1 minute ;
- High - le prix maximum atteint au cours de cette barre 1 minute ;
- Low - le prix minimum atteint au cours de cette barre 1 minute ;
- Close - le prix de fermeture de la barre.

La nouvelle barre 1 minute n'est pas ouverte au moment où une nouvelle minute commence (le nombre de secondes devient égal à 0), mais lorsqu'un tick se produit - un changement de prix d'au moins un point La figure montre la première barre 1 minute de la nouvelle semaine de trading, avec une heure d'ouverture le 10/01/2011 à 00:00. L'écart de prix entre le vendredi et le lundi, que nous voyons sur le graphique, est commun, puisque les taux de change fluctuent, même le week-end, en réponse aux nouvelles informations.

© 2000-2024, MetaQuotes Ltd.

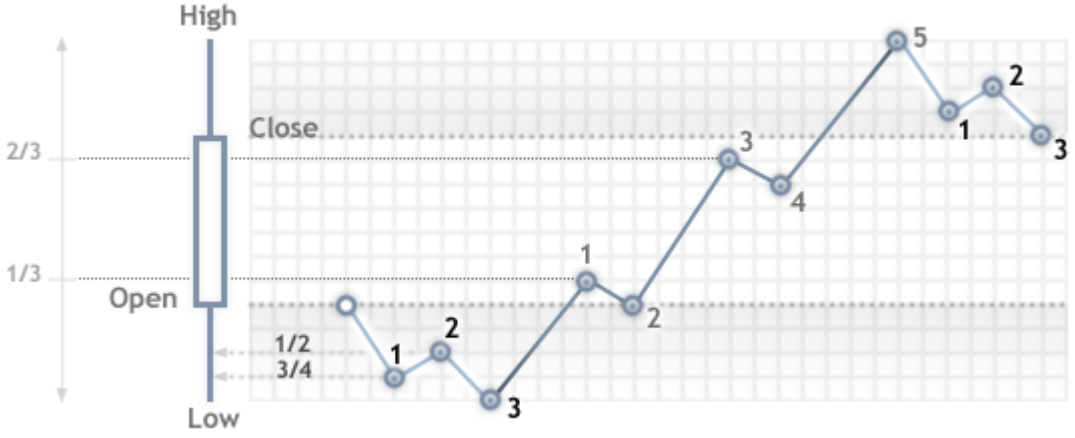


Pour cette barre, nous savons seulement que la barre 1 minute a été ouverte le 10 janvier 2011 à 00 heures 00 minutes, mais on ne sait rien sur les secondes. Elle aurait pu être ouverte à 00:00:12 ou 00:00:36 (12 ou 36 secondes après le début d'un nouveau jour) ou à tout autre moment pendant cette minute. Mais nous savons que le prix d'ouverture de l'EURUSD était à 1,28940 au moment de l'ouverture de la nouvelle barre 1 minute.

De la même manière, nous ne savons pas (à la seconde près) quand le tick correspondant au prix de la clôture de la barre 1 minute examinée a été reçu. Nous ne savons qu'une chose - le dernier prix de Clôture de la barre 1 minute. Pour cette minute, le prix était de 1,28958. Les moments d'apparition des prix High et Low sont également inconnus, mais on sait que les prix maximum et minimum étaient respectivement sur les niveaux 1,28958 et 1,28940.

Pour tester la stratégie de trading, nous avons besoin d'une séquence de ticks sur laquelle le travail de l'Expert Advisor sera simulé. Ainsi, pour chaque barre 1 minute, nous connaissons les **4 points de contrôle** où le prix a été de façon sûre. Si une barre ne dispose que de 4 ticks, nous avons alos suffisamment d'informations pour effectuer un test, mais en général le volume des ticks est supérieur à 4. Par conséquent, il est nécessaire de générer des points de contrôle supplémentaires pour les ticks qui ont eu lieu entre les prix Open, High, Low et Close. Le principe de la génération des ticks en mode "Tous les ticks" est décrit dans l'article Algorithme de génération des ticks dans le Strategy Tester du Terminal MetaTrader 5, dont la figure est présentés plus bas.

© 2000-2024, MetaQuotes Ltd.



Lors du test en mode "Tous les ticks", la fonction OnTick() de l'expert sera **appelée à chaque point de contrôle**. Chaque point de contrôle correspond à un tick de la séquence générée. L'expert recevra l'heure et le prix du tick modélisé de la meme façon qu'il le recevrait en temps réel.

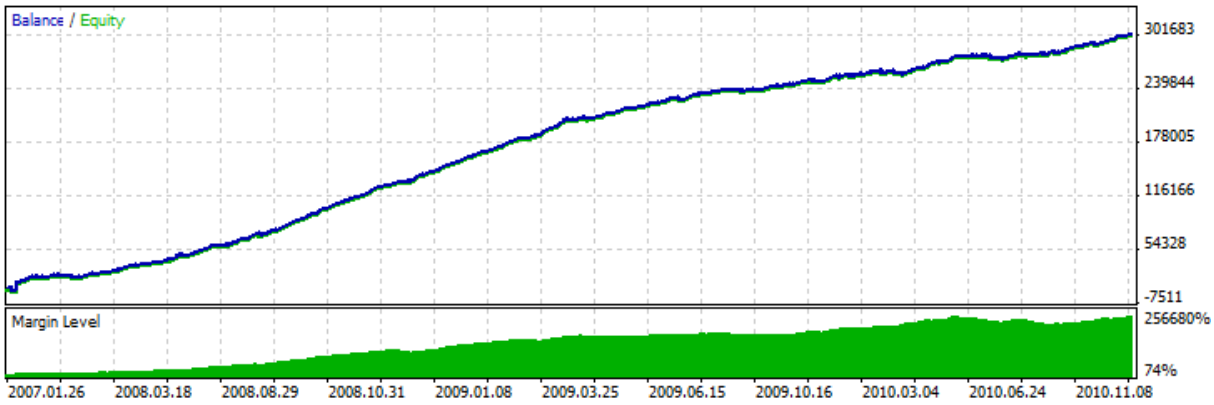
> **Important :** le mode de test "Tous les ticks" est le plus précis, mais il prend beaucoup plus de temps. Pour un test initial de la plupart des stratégies de trading, il suffit d'utiliser l'un de deux autres modes de test.

1 minute OHLC

Le mode "Tous les ticks" est le plus précis des trois modes, mais c'est aussi le plus lent. Le lancement du gestionnaire OnTick() se produit à chaque tick, tandis que le volume des ticks peut être assez important. Pour les stratégies pour lesquelles la séquence des mouvements des prix dans la barre n'est pas importante, il existe un mode de simulation plus rapide et moins fin - "1 minute OHLC".

En mode "1 minute OHLC", la succession des ticks est construite seulement selon les **prix OHLC des barres 1 minute**, le nombre de points de contrôle générés diminue beaucoup - donc, diminue le temps du test. L'appel à la fonction OnTick() est effectué sur tous les points de contrôle, qui sont construits aux prix OHLC des barres 1 minute.

Le refus de génération des ticks supplémentaires intermédiaires entre les prix Open, High, Low et Close amène à l'apparition d'une détermination rigide du développement des prix à partir du moment où le prix Open est déterminé. Cela permet de créer un "Test du Graal", montrant un beau graphique haussier de la balance du test. L'exemple d'un tel Graal est présenté dans le Code Base - Grr-al.



© 2000-2024, MetaQuotes Ltd.

La figure montre un graphique très attractif pour le test de cet EA. Comment a t'il été obtenu ? Nous connaissons 4 prix pour une barre 1 minute et nous savons aussi que le premier est le prix d'Ouverture, et le dernier est le prix de Clôture. Entre eux, il y a les prix High et Low, et la séquence de leur apparition est inconnue, mais on sait que le prix High est supérieur ou égal au prix Open (le prix Low est inférieur ou égal au prix Open).

Il suffit de définir le moment d'entrée du prix Open et d'analyser le tick suivant pour déterminer quel prix nous avons en ce moment - soit High ou Low. Si le prix est inférieur au prix Open, alors nous avons un prix Low - achetons sur ce tick, le tick suivant correspondra au prix High, sur lequel nous fermons à l'achat et nous ouvrons à la vente. Le tick suivant est le dernier, c'est le prix Close, nous y fermons la vente.

Si, après le prix, nous recevons un tick avec un prix supérieur au prix d'ouverture, la séquence des transactions est inversée . Traitons une barre 1 minute dans ce mode et attendons la suivante. Lors du test de cet EA sur l'historique, tout se passe bien, mais une fois que nous l'éxécutons en ligne, la situation change de façon spectaculaire - la ligne d'équilibre reste stable, mais tend à descendre. Pour révéler cette astuce, il suffit de lancer cet EA en mode "Tous les ticks".

Remarque : si les résultats du test de l'EA dans les modes de tests bruts ("1 minute OHLC" et "Prix d'Ouverture uniquement") semblent trop bon, assurez-vous de le tester en mode "Tous les ticks".

Prix d'ouverture uniquement

Dans ce mode, les ticks sont générés sur la base des prix OHLC de la période sélectionnée pour les tests. La fonction OnTick() de l'Expert Advisor fonctionne uniquement au début de la barre au prix d'Ouverture. Grâce à cette caractéristique, les niveaux stops et les ordres en attente peuvent se déclencher à un prix différent de celui spécifié (en particulier lors du test sur des périodes plus élevées). Au lieu de cela, nous avons la possibilité d'exécuter rapidement un test d'évaluation de l'Expert Advisor.

Les périodes W1 et MN1 sont des exceptions dans le mode de génération "Prix d'Ouverture Uniquement" : pour ces périodes, les ticks sont générés pour les prix OHLC de chaque jour, et non pas les prix OHLC de la semaine ou du mois.

Supposons que nous testons un Expert Advisor sur EURUSD H1 en mode "Prix d'Ouverture Uniquement". Dans ce cas, le nombre total de ticks (points de contrôle) ne sera pas supérieur à 4*le nombre de barres d'une heure dans l'intervalle testé. Mais **le gestionnaire OnTick() est appelé uniquement à l'ouverture** de la barre**d'une heure** . Les contrôles requis pour un test correct se produisent sur le reste des ticks (qui sont "cachés" pour l'EA) :

- le calcul des exigences de marge ;
- le déclenchement des Stop Loss et Take Profit ;
- le déclenchement des ordres en cours ;
- la suppression des ordres en attente et expirés.

S'il n'y a pas de position ouverte ou d'ordre en cours, on n'a pas besoin d'effectuer ces contrôles sur les ticks, ce qui augmente considérablement la vitesse. Ce mode "Prix d'Ouverture uniquement" est bien adapté pour le test de stratégies qui n'effectuent des transactions qu'à l'ouverture d'une barre et qui n'utilisent pas d'ordres en attente, d'ordres StopLoss et TakeProfit. Pour de telles stratégies, la précision nécessaire des tests est préservée.

© 2000-2024, MetaQuotes Ltd.

Utilisons l'EA Moving Average du package standard comme exemple d'un EA pouvant être testé dans tous les modes. La logique de cet EA est que toutes les décisions sont prises à l'ouverture de la barre, et les transactions sont effectuées immédiatement, sans utilisation des ordres en attente. Exécutez un test de l'EA sur EURUSD H1 sur un intervalle du 01/09/2010 au 31/12/2010, et comparez les graphiques. La figure montre le graphique du solde du rapport de test pour l'ensemble des trois modes.

Comme vous le voyez, les graphiques sur les différents modes de test sont absolument identiques pour l'EA Moving Average du package standard.

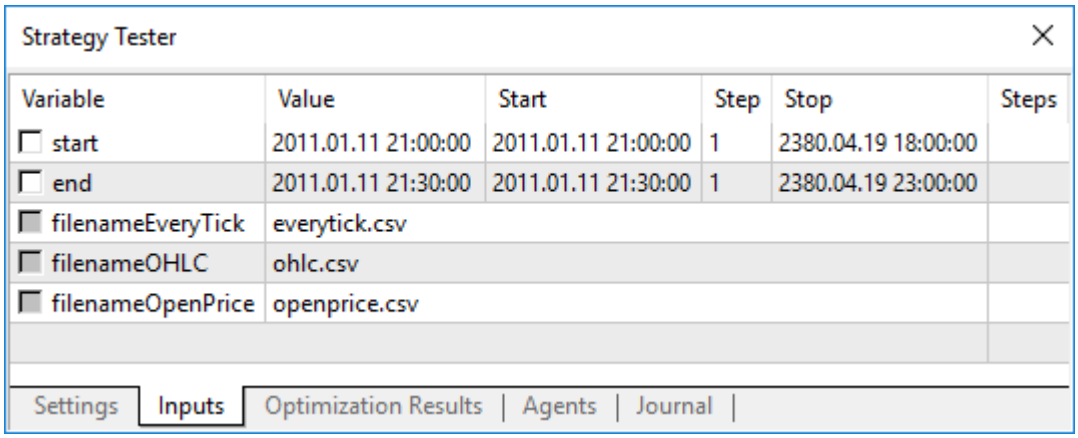
Il y a quelques limitations sur le mode "Prix d'Ouverture uniquement" :

- On ne peut pas utiliser le mode d'exécution "Délai Aléatoire" ;
- Dans l'expert testé, vous ne pouvez pas accéder aux données de la période inférieure à celle utilisée pour les tests/l'optimisation. Par exemple, si vous exécutez les tests/optimisation sur la période H1, vous pouvez accéder aux données de H2, H3, H4, etc., mais pas à celles de M30, M20, M10, etc. En outre, les périodes les plus élevées utilisées doivent être des multiplies de la période de test. Par exemple, si vous exécutez les tests en M20, vous ne pouvez pas accéder aux données de M30, mais il est possible d'accéder à H1. Ces limitations sont liées à l'impossibilité d'obtenir des données des périodes inférieures ou non-multiples des barres générées pendant le test/optimisation.
- Les limitations sur l'accès aux données d'autres périodes sont également applicables aux autres symboles dont les données sont utilisées par l'Expert Advisor. Dans ce cas, la limitation pour chaque symbole dépend de la première période consultée lors du test/optimisation. Supposons, au cours des essais sur EURUSD H1, qu'un Expert Advisor accède aux données de GBPUSD en M20. Dans ce cas, l'Expert Advisor sera en mesure d'utiliser d'autres données de EURUSD en H1, H2, etc., ainsi que la paire GBPUSD en M20, H1, H2, etc.

Remarque : le mode "Prix d'Ouverture uniquement" est le plus rapide pour les tests, mais ne convient pas à toutes les stratégies de trading. Sélectionnez le mode de test désiré sur la base des caractéristiques du système de trading.

© 2000-2024, MetaQuotes Ltd.

Pour conclure la section sur les modes de génération des ticks, comparons les différents modes de génération des ticks pour l'EURUSD, pour deux barres M15 sur l'intervalle du 11/01/2011 à 21:00:00 au 11/01/2011 à 21:30:00. Les ticks sont enregistrés dans différents fichiers en utilisant l'EA WriteTicksFromTester.mq5 et la fin de ces noms de fichiers sont spécifiés dans les paramètres d'entrée filenamEveryTick, filenameOHLC et filenameOpenPrice.

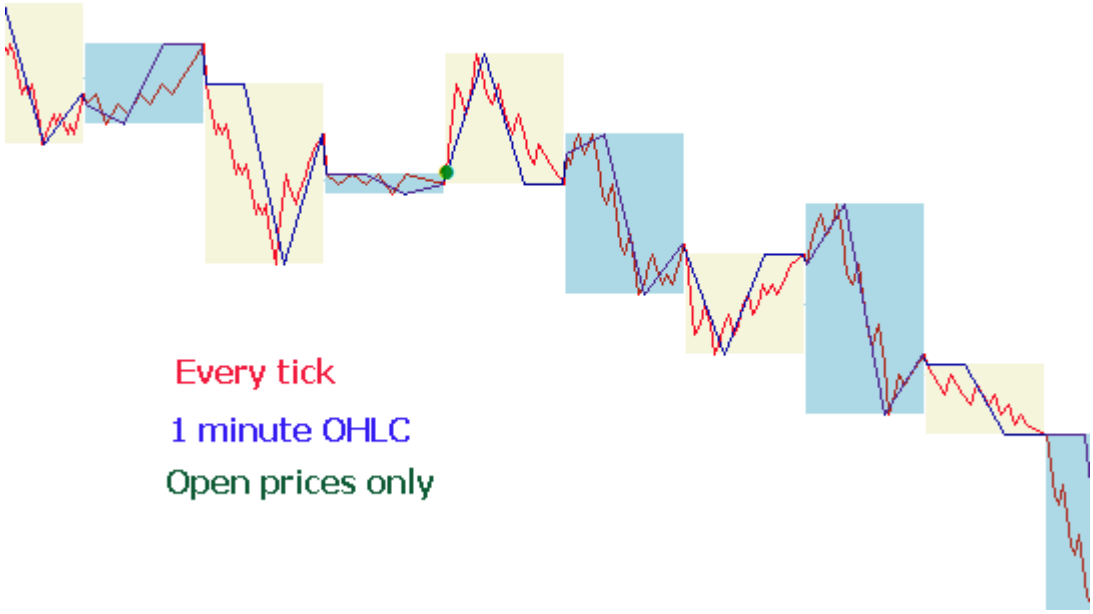


Strategy Tester

Variable	Value	Start	Step	Stop	Steps
start	2011.01.11 21:00:00	2011.01.11 21:00:00	1	2380.04.19 18:00:00	
end	2011.01.11 21:30:00	2011.01.11 21:30:00	1	2380.04.19 23:00:00	
filenameEveryTick	everytick.csv				
filenameOHLC	ohlc.csv				
filenameOpenPrice	openprice.csv				

Settings | Inputs | Optimization Results | Agents | Journal

Pour obtenir trois fichiers avec trois séquences de tick (pour chacun des modes suivants : "Tous les ticks", "1 minute OHLC" et "Prix d'Ouverture Uniquement"), l'EA a été lancé trois fois dans les modes correspondants, en exécution simple. Ensuite, les données de ces trois fichiers sont affichées sur le graphique en utilisant l'indicateur TicksFromTester.mq5. Le code de l'indicateur est joint au présent article.



Par défaut, toutes les opérations sur les fichiers dans le langage MQL5 sont effectuées dans la "sandbox" et pendant le test, l'EA n'a accès qu'à sa propre "sandbox de fichiers". Pour que l'indicateur et l'EA puissent travailler avec les fichiers d'un dossier lors de tests, nous avons utilisé le flag FILE_COMMON. Exemple de code de l'EA :

© 2000-2024, MetaQuotes Ltd.

```
//--- ouverture du fichier
   file=FileOpen(filename,FILE_WRITE|FILE_CSV|FILE_COMMON,";");
//--- vérification du succès de l'opération
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Erreur dans l'ouverture du fichier pour l'enregistrement %s. Le coc
      return;
     }
   else
     {
      //--- affiche le nom du dossier
      PrintFormat("Le fichier sera enregistré dans le dossier %s",TerminalInfoString(T
     }
```

Pour lire les données de l'indicateur, nous avons également utilisé le flag FILE_COMMON, cela a permis d'éviter le transfert des fichiers nécessaires manuellement d'un dossier à l'autre.

```
//--- ouverture du fichier
   int file=FileOpen(fname,FILE_READ|FILE_CSV|FILE_COMMON,";");
//--- vérification du succès de l'opération
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Erreur dans l'ouverture du fichier pour la lecture %s. Le code de l
      return;
     }
   else
     {
      //--- affichage du nom du dossier
      PrintFormat("Le fichier sera lu du dossier %s",TerminalInfoString(TERMINAL_COMMO
     }
```

Simulation du spread

La différence entre les prix Bid et Ask s'appelle le spread. Pendant le test, le spread n'est pas modelisé, mais est pris à partir des données historiques. Si le spread est inférieur ou égal à zéro dans les données historiques, le dernier spread connu (au moment de la génération) est utilisé par l'agent de test.

Dans le Strategy Tester, le spread est toujours considéré comme flottant. C'est à dire que SymbolInfoInteger(symbol, SYMBOL_SPREAD_FLOAT) retourne toujours vrai.

En outre, les données historiques contiennent les valeurs des ticks et les volumes des transactions. Pour le stockage et la récupération des données, nous utilisons une structure spéciale MqlRates:

```
struct MqlRates
  {
   datetime time;         // l'heure de l'ouverture de la barre
   double   open;         // le prix de l'ouverture Open
   double   high;         // le prix le plus haut High
   double   low;          // le prix le plus bas Low
   double   close;        // le prix de la clôture Close
   long     tick_volume;  // le volume du tick
   int      spread;       // le spread
   long     real_volume;  // le volume réel
  };
```

Utiliser des ticks réels pendant un test

© 2000-2024, MetaQuotes Ltd.

Les tests et les optimisations sur des ticks réels sont aussi proches des conditions réelles que possible. Au lieu de générer des ticks basées sur les données des barres 1 minutes, il est possible d'utiliser les ticks réels accumulés par un courtier. Ce sont les ticks issus des places boursières et des fournisseurs de liquidité.

Pour assurer la meilleure précision des tests, les barres 1 minute sont également utilisées dans le mode des ticks réels. Les barres sont appliquées pour vérifier et corriger les données des ticks. Ceci vous permet également d'éviter une divergence des graphiques dans le tester et dans le terminal client.

Le tester compare les données du tick avec les paramètres de la barre 1 minute : un tick ne doit pas être supérieur ou inférieur aux niveaux High et Low, le tick initial et le tick final doivent également coincider avec les prix Open/Close de la barre. Le volume est aussi comparé. Si un décalage est détecté, tous les ticks correspondant à cette barre 1 minute sont rejetés. Les ticks générés sont utilisés à leur place (comme dans le mode "Chaque tick").

Si l'historique d'un symbole possède une barre 1 minute sans données de tick, le tester génère des ticks dans le mode "Chaque tick". Cela permet de tracer un graphique correct dans le tester au cas où les données d'un courtier sont insuffisantes.

Si l'historique d'un symbole n'a pas de barre 1 minute, mais que les données du tick correspondant à cette minute sont présentes, les données peuvent être utilisées dans le tester. Par exemple, les paires de devises sont formées en utilisant les prix Last. Si seulement des ticks avec les prix Bid/Ask mais sans le prix Last arrivent du serveur, la barre n'est pas générée. Le tester utilise ces données puisqu'elles ne contredisent pas les données en minute.

Les données du tick peuvent ne pas coïncider avec les barres 1 minute pour diverses raisons, par exemple en raison de pertes de connexion ou d'autres défaillances lors de la transmission des données d'une source jusqu'au terminal client. Les données 1 minute sont considérées comme plus fiables au cours des tests.

Gardez en mémoire les éléments suivants lors de tests sur des ticks réels :

- Lors du lancement d'un test, les données 1 minute d'un symbole sont synchronisées avec le premier tick.
- Les ticks sont stockés dans le cache du symbole du strategy tester. La taille du cache ne doit pas excéder 128 000 ticks. Lorsque de nouveaux ticks arrivent, les données les plus anciennes sont supprimées du cache. Cependant, la fonction CopyTicks permet de récupérer les ticks en dehors du cache (seulement pour des tests sur des ticks réels).Dans ce cas, les données sont récupérées depuis la base de données des ticks du tester qui est totalement similaire à la base de données correspondante du terminal client. Aucune correction de barre 1 minute n'est faite dans cette base. De plus, les ticks peuvent différer de ceux stocker dans le cache.

Les Variables globales du terminal client

Pendant le test, les variables globales du terminal client sont également émulées, mais elles ne sont aucunement liées aux variables globales actuelles du terminal, qui peuvent être vues dans le terminal en utilisant la touche F3. Cela signifie que toutes les opérations avec les variables globales du terminal, au cours des tests, ont lieu à l'extérieur du terminal client (dans l'agent de test).

Calcul des indicateurs au cours des tests

© 2000-2024, MetaQuotes Ltd.

En mode temps réel, les valeurs des indicateurs sont calculées à chaque tick.

Dans le Strategy Tester, les indicateurs sont calculés uniquement lorsqu'ils sont consultés pour les données, c'est-à-dire lorsque des valeurs du buffer de l'indicateur sont demandées. Les seules exceptions sont les indicateurs personnalisés avec la propriété #property tester_everytick_calculatespécifiée. Dans ce cas, le recalcul est effectué à chaque tick.

En mode de test visuel, tous les indicateurs sont inconditionnellement recalculés à l'arrivée d'un nouveau tick afin d'être correctement affichés sur le graphique de test visuel.

L'indicateur est calculé une fois par tick. Toutes les demandes ultérieures de données de l'indicateur n'entraînent pas de recalcul jusqu'à ce qu'un nouveau tick arrive. Par conséquent, si la minuterie est activée dans un EA via la fonction EventSetTimer(), les données de l'indicateur sont demandées à partir du dernier tick avant chaque appel du gestionnaire OnTimer(). Si l'indicateur n'a pas encore été calculé sur le dernier tick, les calculs des valeurs de l'indicateur sont lancés. Si les données ont déjà été préparées, elles sont fournies sans nouveau recalcul.

Ainsi, tous les calculs d'indicateurs sont effectués de la manière la plus économe en ressources — si l'indicateur a déjà été calculé à un tick donné, ses données sont fournies "telles quelles". Aucun recalcul n'est lancé.

Chargement de l'historique au cours des tests

L'historique d'un symbole à tester est synchronisé et chargé par le terminal à partir du serveur de trading avant de commencer le processus de test. La première fois, le terminal charge l'historique disponible du symbole afin de ne pas le demander plus tard. Par la suite, sules les nouvelles données sont chargées.

Un agent de test reçoit l'historique d'un symbole à tester à partir du terminal client juste après le lancement du test. Si les données d'autres instruments sont utilisées dans le processus de test (par exemple, Expert Advisor multidevises), l'agent de test demande l'historique nécessaire à partir du terminal client lors du premier appel à ces données. Si les données historiques sont disponibles dans le terminal, elles sont immédiatement transmises à l'agent de test. Si les données ne sont pas disponibles, le terminal les demande et les télécharge à partir du serveur, puis les passe à l'agent de test.

Les données des instruments supplémentaires sont également nécessaires pour le calcul des taux croisés pour les opérations de trading. Par exemple, lors du test d'une stratégie sur l'EURCHF avec la monnaie de dépôt en USD, avant le traitement de la première opération de trading, l'agent de test demande l'historique des données de la paire EURUSD et USDCHF au terminal client, bien que la stratégie ne contienne pas l'utilisation directe de ces symboles.

Avant de tester une stratégie multi-devises, il est recommandé de télécharger toutes les données historiques nécessaires pour le terminal client. Cela permettra d'éviter des retards dans les tests/optimisation dûs au téléchargement des données requises. Vous pouvez télécharger l'historique, par exemple, en ouvrant les graphiques appropriés et les faire défiler vers le début de l'historique. Un exemple du chargement forcé de l'historique dans le terminal est disponible dans la section MQL5 Organisation de l'Accès aux Données.

Les agents de test reçoivent à leur tour l'historique du terminal. Lors du test suivant, le testeur ne charge pas l'historique du terminal, puisque les données nécessaires sont disponibles depuis l'exécution précédente du testeur.

© 2000-2024, MetaQuotes Ltd.

- Le terminal ne charge l'historique du serveur qu'une seule fois au premier appel de l'agent par le terminal pour recevoir l'historique du symbole testé. L'historique est chargé sous forme d'un paquet pour réduire le trafic.
- Les ticks ne sont pas envoyés sur le réseau, ils sont générés sur les agents de test.

Tests Multi-Devises

Le Strategy Tester nous permet de tester des stratégies tradant sur plusieurs symboles. Ces EA sont habituellement appelés Expert Advisors multi-devises, puisqu'à l'origine, dans les plateformes précédentes, les tests n'étaient effectués que pour un seul symbole. Dans le Strategy Tester du terminal MetaTrader 5, nous pouvons modéliser le trading pour tous les symboles disponibles.

Le testeur charge l'historique des symboles utilisés à partir du **terminal client** (pas à partir du serveur de trading !) automatiquement lors du premier appel des données du symbole.

L'agent de test télécharge seulement l'historique manquant, avec une petite marge pour fournir les données nécessaires, pour le calcul des indicateurs à l'heure du début des tests. Pour les périodes D1 et inférieures, le volume minimum de l'historique téléchargé est de un an. Ainsi, si nous lançons un test sur l'intervalle 01/11/2010 - 01/12/2010 (test pour un intervalle d'un mois) avec une période de M15 (chaque barre est égale à 15 minutes), le terminal demandera l'historique de l'instrument pour l'ensemble de l'année 2010. Pour les périodes Hebdomadaires, nous allons demander un historique de 100 barres, environ deux ans (52 semaines par an). Pour un test sur une période Mensuelle, l'agent demandera l'historique sur 8 ans (12 mois x 8 ans = 96 mois).

S'il n'y a pas la quantité nécessaires de barres, **la date de début sera automatiquement déplacée** vers le passé pour fournir la quantité de barres nécessaire avant le test.

Pendant le test, le "Market Watch" est également émulé, à partir duquel on peut obtenir des informations sur les symboles. Par défaut, au début du test, il n'y a qu'un symbole dans le "Market Watch" du Strategy Tester - le symbole sur lequel le test est exécuté. Tous les symboles nécessaires sont connectés au "Market Watch" du Strategy Tester (pas à celui du terminal !) automatiquement lorsqu'ils sont appelés.

Avant de commencer le test d'un Expert Advisor multi-devises, il est nécessaire de sélectionner les symboles requis dans le "Market Watch" du terminal et de charger les données requises Lors du premier appel d'un symbole "étranger", son historique sera automatiquement synchronisé entre l'agent de test et le terminal client. Un symbole "étranger" est un symbole autre que celui sur lequel le test est en cours d'exécution.

Le renvoi aux données d'un "autre" symbole se produit dans les cas suivants :

- lorsque vous utilisez la fonction et IndicatorCreate() des indicateurs techniques sur le symbole/la période ;
- La demande de données au "Market Watch" pour l'autre symbole :
  1. SeriesInfoInteger
  2. Barres
  3. SymbolSelect
  4. SymbolIsSynchronized

© 2000-2024, MetaQuotes Ltd.

5. SymbolInfoDouble
6. SymbolInfoInteger
7. SymbolInfoString
8. SymbolInfoTick
9. SymbolInfoSessionQuote
10. SymbolInfoSessionTrade
11. MarketBookAdd
12. MarketBookGet

- la demande de la série chronologique pour un symbole/une période en utilisant les fonctions suivantes :
  1. CopyBuffer
  2. CopyRates
  3. CopyTime
  4. CopyOpen
  5. CopyHigh
  6. CopyLow
  7. CopyClose
  8. CopyTickVolume
  9. CopyRealVolume
  10. CopySpread

Au moment du premier appel à un symbole "autre", le processus de test s'arrête et l'historique est téléchargé pour le symbole/la période à partir du terminal vers l'agent de test. En même temps, la séquence de ticks pour ce symbole est générée.

Une séquence individuelle de ticks est générée pour chaque symbole, selon le mode sélectionné de génération des ticks. Vous pouvez également demander explicitement l'historique des symboles souhaités en appelant la fonction SymbolSelect() dans le gestionnaire OnInit() - le téléchargement de l'historique sera fait immédiatement avant le test de l'Expert Advisor.

Ainsi, pour effectuer des tests multi-devises dans le terminal client MetaTrader 5, aucun effort supplémentaire n'est nécessaire. Il suffit d'ouvrir les graphiques des symboles appropriés dans le terminal client. L'historique sera automatiquement téléchargé à partir du serveur pour tous les symboles nécessaires, à condition qu'il contienne ces données.

Simulation du Temps dans le Strategy Tester

Au cours des tests, l'heure locale TimeLocal() est toujours égale à l'heure du serveur TimeTradeServer(). À son tour, l'heure du serveur est toujours égale à l'heure GMT - TimeGMT(). Ainsi, toutes ces fonctions affichent la même heure lors des tests.

L'absence de différence entre l'heure GMT, l'heure locale, et l'heure du serveur dans le Strategy Tester se fait délibérément dans le cas où il n'y a pas de connexion au serveur. Les résultats des tests doivent toujours être identiques, indépendamment de la présence ou non d'une connexion. L'information sur l'heure du serveur n'est pas stockée localement, et est prise à partir du serveur.

© 2000-2024, MetaQuotes Ltd.

Objets Graphiques pendant le Test

Lors des tests/optimisations, les objets graphiques ne sont pas tracés. Ainsi, en se référant aux propriétés d'un objet créé lors des tests/optimisations, un Expert Advisor recevra des valeurs nulles.

Cette limitation ne concerne pas les tests en mode visuel.

La Fonction OnTimer() dans le Strategy Tester

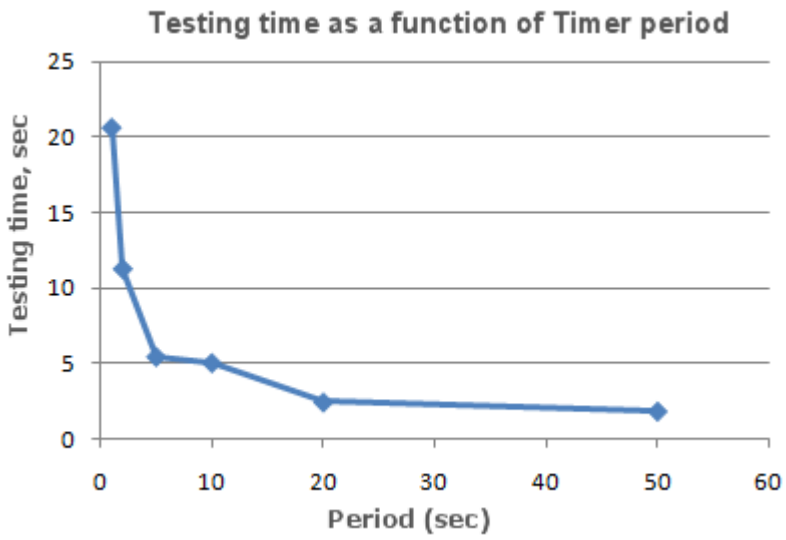
MQL5 prévoit la possibilité de gérer des événements de timer. L'appel du gestionnaire OnTimer() est fait quel que soit le mode de test. Cela signifie que si un test est en cours d'exécution en mode "Prix d'Ouverture Uniquement" pour la période H4, et que l'EA a un timer réglé à un appel par seconde, alors à l'ouverture de chaque barre en H4, le gestionnaire OnTick() sera appelé une seule fois, et le gestionnaire OnTimer() sera appelé 14.400 fois (3.600 secondes * 4 heures). . L'augmentation de temps qui en découlera dépend de la logique de l'EA.

Pour vérifier la dépendance du temps du test de la fréquence donnée du timer, nous avons créé un EA simple, sans aucune opération de trading.

```
//--- paramètres d'entrée
input int      timer=1;              // valeur du time, sec
input bool     timer_switch_on=true; // le timer est activé
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- démarrage du timer si timer_switch_on==true
   if(timer_switch_on)
     {
      EventSetTimer(timer);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- arrêt du timer
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
// ne fait rien, le corps du gestionnaire est vide
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

Les mesures de temps du test ont été faites avec différentes valeurs du paramètre du timer (périodicité de l'événement Timer). Le graphique de la dépendance du temps du test T à la valeur de la périodicité Period est construit sur les données reçues.



Il est clair que, plus le paramètre timer est petit lors de l'initialisation de la fonction EventSetTimer (timer), plus courte sera la période (Period) entre les appels du gestionnaire OnTimer(), et plus grand est le test du temps T, dans les mêmes autres conditions.

Fonction Sleep() dans le Strategy Tester

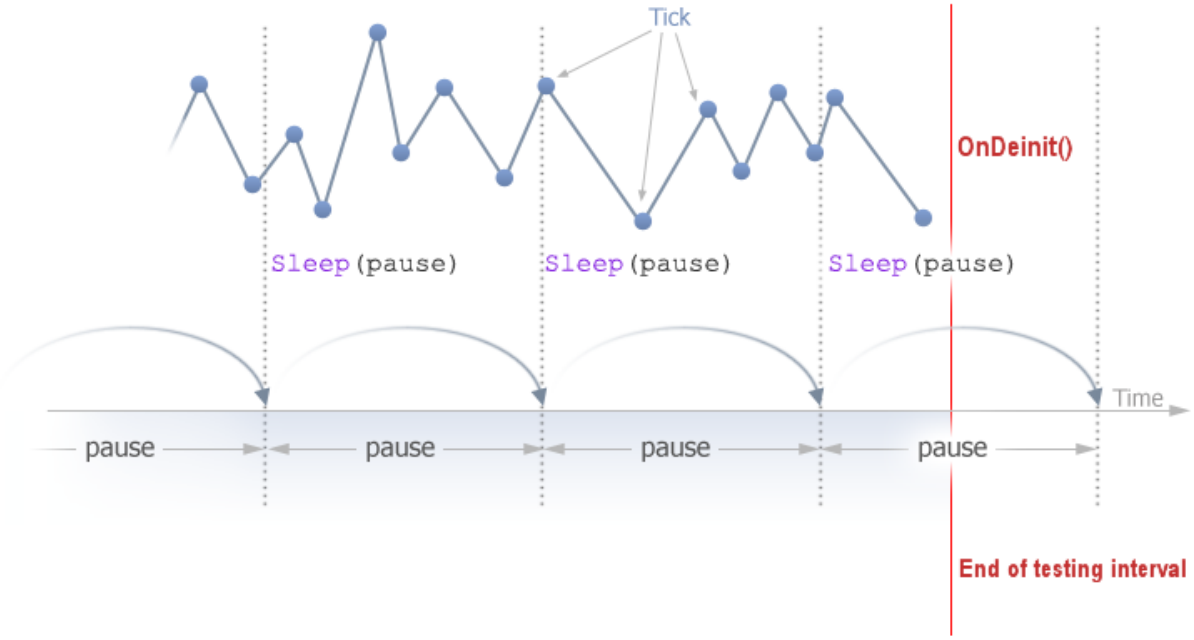
La fonction Sleep() permet à l'EA ou à un script de suspendre l'exécution du programme MQL5 pendant un certain temps, lorsque l'on travaille sur le graphique. Cela peut être utile lors de la demande de données, qui ne sont pas prêtes au moment de la demande et qu'il est nécessaire d'attendre jusqu'à ce qu'elles le soient. Un exemple détaillé de l'utilisation de la fonction Sleep() peut être trouvé dans la section Organisation de l'Accès aux Données.

Dans le testeur, les appels Sleep() ne bloquent pas le déroulement du test. A l'appel de Sleep(), les ticks générés sont "joués" dans le délai spécifié, ce qui peut entraîner le déclenchement des ordres en attente, stops, etc. Après l'appel de Sleep(), le temps modelisé dans le testeur augmente de l'intervalle indiqué dans le paramètre de la fonction Sleep.

Si, à la suite de l'exécution de la fonction Sleep(), l'heure actuelle dans le testeur dépasse l'intervalle du test, l'erreur "Boucle infinie détectée dans Sleep". Si vous recevez cette erreur, les résultats des tests ne sont pas rejetés, tous les calculs sont effectués en totalité (le nombre de transactions, l'affaissement, etc.), et les résultats de ce test sont transmis au terminal.

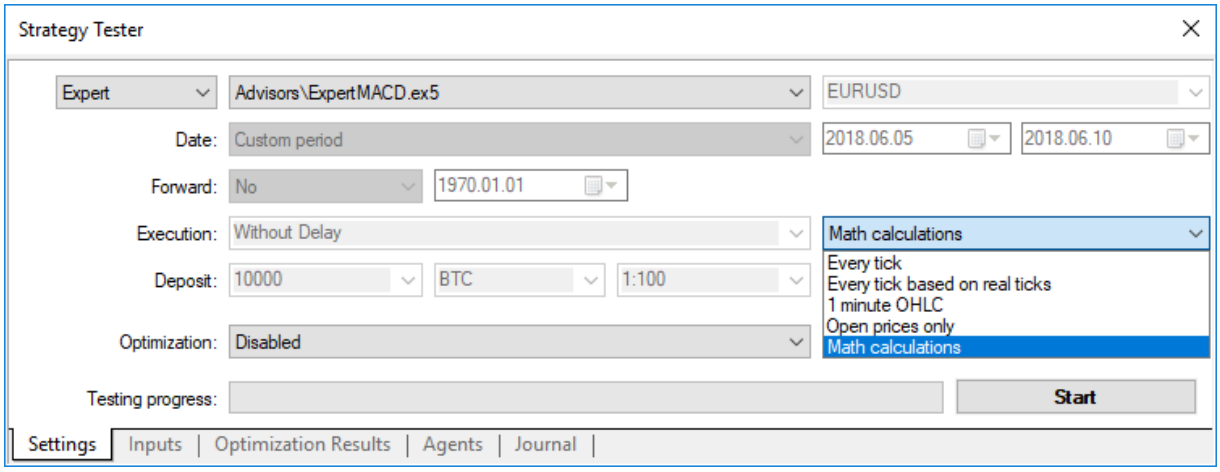
La fonction Sleep() ne fonctionne pas dans OnDeinit(), puisque après son appel l'heure du test se trouve en dehors de l'intervalle du test de façon certaine.

© 2000-2024, MetaQuotes Ltd.



Utilisation du Strategy Tester pour l'Optimisation des Problèmes dans les Calculs Mathématiques

On peut utiliser le testeur dans le terminal MetaTrader 5 non seulement pour vérifier des stratégies de trading, mais également pour des calculs mathématiques. Pour l'utiliser, il est nécessaire de sélectionner le mode "Calculs Mathématiques" :



Dans ce cas, seules trois fonctions seront appelées : OnInit(), OnTester(), OnDeinit(). En mode "Calculs Mathématiques", le Strategy Tester ne génère pas de ticks et télécharge l'historique. Dans ce cas, seules trois fonctions seront appelées : OnInit(), OnTester(), OnDeinit().

Si la date de fin du test est inférieure ou égale à la date de début du test, cela signifiera que le test sera exécuté en mode "Calculs Mathématiques".

Lorsque vous utilisez le testeur pour résoudre des problèmes mathématiques, l'ajout de l'historique et la génération de ticks ne sont pas effectués.

© 2000-2024, MetaQuotes Ltd.

Un problème mathématique typique à résoudre dans le Strategy Tester de MetaTrader 5 : la recherche de l'extremum d'une fonction à plusieurs variables. Pour le résoudre, nous devons :

- Placer le bloc de calcul de la valeur de la fonction à plusieurs variables dans la fonction OnTester() et retourner la valeur calculée avec return(la valeur_de la fonction);
- Les paramètres de la fonction doivent être définis comme des variables d'entrée de l'Expert Advisor ;

Pour compiler l'EA, ouvrez la fenêtre du "Strategy Tester". Dans l'onglet "Paramètres d'entrée", sélectionnez les variables d'entrées demandées, et définissez l'ensemble des valeurs des paramètres en spécifiant les valeurs de démarrage, d'arrêt et de pas pour chacune des variables de la fonction.

Sélectionnez le type d'optimisation - "Algorithme Complet Lent" (recherche complète de l'espace des paramètres) ou "Algorithme Génétique Rapide". Pour une simple recherche de l'extremum de la fonction, il est préférable de choisir une optimisation rapide, mais si vous voulez calculer les valeurs pour l'ensemble des variables, alors il est préférable d'utiliser l'optimisation lente.

Sélectionnez le mode "Calculs Mathématiques", utilisez le bouton "Démarrer", et exécutez la procédure d'optimisation. A noter que lors de l'optimisation, le Strategy Tester recherche les valeurs maximum de la fonction OnTester. Pour trouver un minimum local, retournez l'inverse de la valeur de la fonction calculée à partir de la fonction OnTester :

```
return(1/valeur_de la fonction);
```

Il est nécessaire de vérifier que valeur_de_la_fonction n'est pas égale à zéro, autrement une erreur critique de division par zéro peut être générée. Il existe une autre variante, plus facile et qui ne fausse pas les résultats de l'optimisation ; elle a été suggérée par les lecteurs de cet article :

```
return(-valeur_de la fonction);
```

Dans cette variante, il n'est pas utile vérifier si valeur_de_la_fonction est égal à zéro et la surface des résultats de l'optimisation dans la représentation 3D a la même forme. La seule différence est qu'elle est inversée symétriquement par rapport à l'initiale.

A titre d'exemple, nous fournissons la fonction sink() :

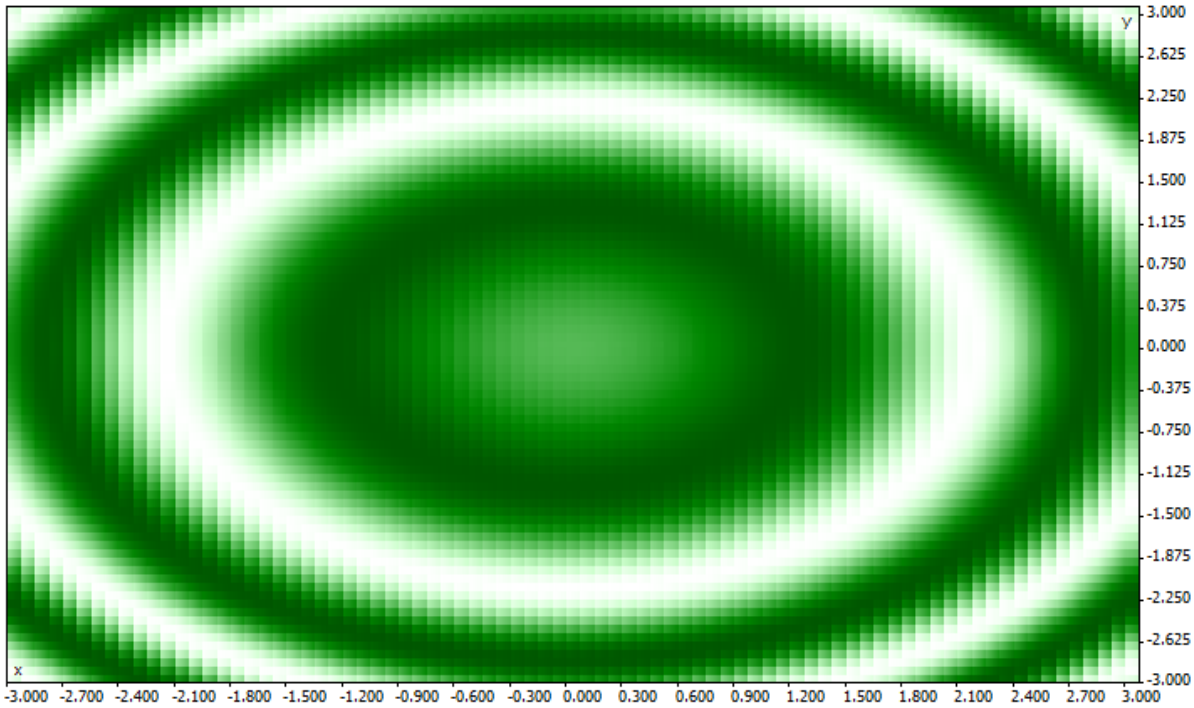
$$sink(x,y) = \sin\left(x^2 + y^2\right)$$

Plaçons le code de l'EA pour trouver l'extremum de cette fonction dans la fonction OnTester() :

© 2000-2024, MetaQuotes Ltd.

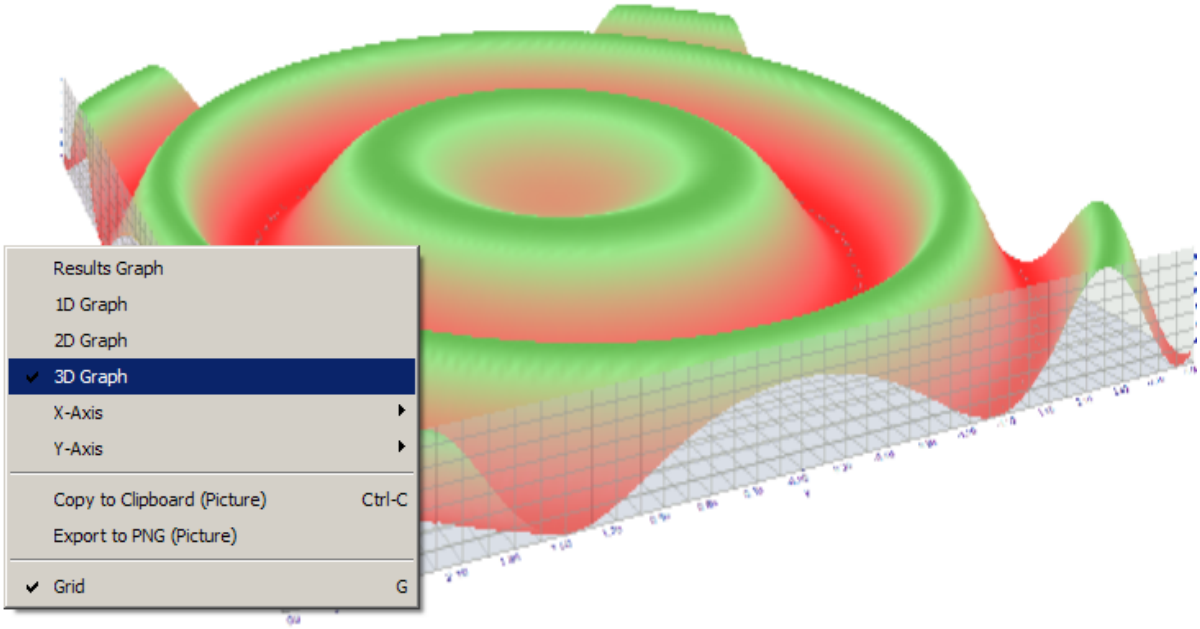
```
//+------------------------------------------------------------------+
//|                                                        Sink.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- paramètres d'entrée
input double   x=-3.0; // start=-3, step=0.05, stop=3
input double   y=-3.0; // start=-3, step=0.05, stop=3
//+------------------------------------------------------------------+
//| Fonction du tester                                               |
//+------------------------------------------------------------------+
double OnTester()
  {
//---
   double sink=MathSin(x*x+y*y);
//---
   return(sink);
  }
//+------------------------------------------------------------------+
```

Effectuons une optimisation et voyons les résultats de l'optimisation sous la forme d'un graphique 2D.



Meilleure est la valeur pour une paire de données des paramètres (x, y), plus la couleur est saturée. Comme il était attendu de la vue de la formule de la fonction sink(), ses valeurs forment des cercles concentriques avec un centre au point (0,0). On peut voir dans le graphique 3D que la fonction sink() n'a pas d'extremum absolu. Graphique 3D de la fonction Sink()

© 2000-2024, MetaQuotes Ltd.



Synchronisation des Barres en mode "Prix d'Ouverture Uniquement"

Le testeur dans le terminal client MetaTrader 5 permet de vérifier les fameux EA "multi-devises". Un EA multi-devises est un EA qui trade sur 2 symboles ou plus.

Le test des stratégies tradant sur plusieurs symboles impose quelques exigences techniques supplémentaires sur le testeur :

- la génération des ticks pour ces symboles ;
- le calcul des valeurs des indicateurs pour ces symboles ;
- le calcul des exigences de marge pour ces symboles ;
- la synchronisation des séquences de tick générées pour tous les symboles.

Le Strategy Tester produit et joue une séquence de ticks pour chaque instrument selon le mode de trading sélectionné. En même temps, une nouvelle barre est ouverte pour chaque symbole, indépendamment de la façon dont la barre a été ouverte sur un autre symbole. Cela signifie que lors du test d'un expert multi-devises, une situation peut se produit (et se produit souvent) lorsque une nouvelle barre est ouverte sur un instrument, mais pas encore sur l'autre instrument. Ainsi, pendant le test tout se passe comme dans la réalité.

Cette simulation authentique de l'historique dans le testeur ne pose pas de problème tant que les modes "Chaque tick" et "1 minute OHLC" sont utilisés. Pour ces modes, suffisamment de ticks sont générés pour une bougie pour pouvoir attendre que la synchronisation des barres des différents symboles soit effectuée. Mais comment tester les stratégies multi-devises en mode "Prix d'Ouverture Uniquement" si la synchronisation des barres sur les instruments de trading est obligatoire ? Dans ce mode, l'EA est n'est appelé que sur un tick, correspondant à l'heure d'ouverture des barres.

Nous allons l'expliquer avec un exemple : nous testons un expert sur le symbole EURUSD, et une nouvelle bougie en H1 a été ouverte sur l'EURUSD. Il est aisé de reconnaître ce fait - pendant le test en mode "Prix d'Ouverture Uniquement", l'événement NewTick correspond au moment de l'ouverture de la

© 2000-2024, MetaQuotes Ltd.

barre sur la période testée. Mais il n'y a aucune garantie qu'une nouvelle bougie a été ouverte sur le symbole USDJPY, qui est utilisé dans l'expert.

Dans des conditions ordinaires, il suffit de terminer le travail de la fonction OnTick() et de vérifier l'apparition d'une nouvelle barre USDJPY sur le tick suivant. Mais pendant le test en mode "Prix d'ouverture uniquement", il n'y aura pas d'autres ticks, et il peut donc sembler que ce mode ne convient pas pour tester des EA multi-devises. Mais ce n'est pas vrai - n'oubliez pas que le testeur sur MetaTrader 5 se comporte comme dans la vraie vie. On peut attendre jusqu'à ce qu'une nouvelle barre s'ouvre sur un autre symbole en utilisant la fonction Sleep() !

Le code de l'EA Synchronize_Bars_Use_Sleep.mq5 montre un exemple de synchronisation des barres dans le mode "Prix d'Ouverture Uniquement" :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                   Synchronize_Bars_Use_Sleep.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- paramètres d'entrée
input string   other_symbol="USDJPY";
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vérification du symbole courant
   if(_Symbol==other_symbol)
     {
      PrintFormat("Il est nécessaire d'indiquer un autre symbole ou lancer le test su
      //--- fin du test de l'expert
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction tick de l'expert                                        |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- variable statique pour stocker l'heure d'ouverture de la dernière barre
   static datetime last_bar_time=0;
//--- signe indiquant que l'heure d'ouverture de la dernière barre sur les symboles d
   static bool synchonized=false;
//--- si la variable statique n'est pas encore initialisée
   if(last_bar_time==0)
     {
      //--- c'est le premier appel, stocke l'heure de l'ouverture et sort
      last_bar_time=(datetime)SeriesInfoInteger(_Symbol,Period(),SERIES_LASTBAR_DATE);
      PrintFormat("La variable last_bar_time est initialisée avec la valeur %s",TimeT
     }
//--- récupère l'heure d'ouverture de la dernière barre selon le symbole
   datetime curr_time=(datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DA
//--- si l'heure d''ouverture de la barre actuelle ne correspond pas à celle qui est
   if(curr_time!=last_bar_time)
     {
      //--- stocke l'heure d'ouverture de la nouvelle barre dans la variable statique
      last_bar_time=curr_time;
      //--- la synchronisation est rompue, le flag est mis à false
      synchonized=false;
      //--- construction du message sur cet événement
      PrintFormat("Une nouvelle barre s'est ouverte sur le symbole %s dans %s",_Symbo
     }
//--- stockera l'heure d''ouverture de la barre sur le symbole étranger
   datetime other_time;
//--- boucle jusqu'à ce que l'heure d'ouverture de la dernière barre sur un autre sym
   while(!(curr_time==(other_time=(datetime)SeriesInfoInteger(other_symbol,Period(),S
     {
      PrintFormat("Attendons 5 secondes..");
      //--- attente de 5 secondes et demande encore SeriesInfoInteger(other_symbol,Pe
      Sleep(5000);
```

```
      }
//--- l'heure d'ouverture de la barre est identique maintenant pour les deux symboles
   synchonized=true;
   PrintFormat("L'heure d'ouverture de la dernière barre sur le symbole %s: %s",_Symbo
   PrintFormat("L'heure d'ouverture de la dernière barre sur le symbole %s: %s",other_
//--- TimeCurrent() ne convient pas, utilise TimeTradeServer() pour
   Print("Les barres étaient synchronisées dans ",TimeToString(TimeTradeServer(),TIME_
  }
//+------------------------------------------------------------------+
```

Prêtez attention à la dernière ligne de l'EA, qui affiche l'heure actuelle quand la synchronisation a été établie :

```
   Print("Les barres étaient synchronisées dans ",TimeToString(TimeTradeServer(),TIME_
```

Pour afficher l'heure actuelle, nous avons utilisé la fonction TimeTradeServer(), plutôt queTimeCurrent(). La fonction TimeCurrent () retourne l'heure du dernier tick, qui n'a pas changé après l'utilisation de la fonction Sleep(). Exécutez l'EA en mode "Prix d'Ouverture Uniquement" et vous verrez un message sur la synchronisation des barres.

Core 1	2010.12.01 20:00:05	The bars are synchronized at 2010.12.01 20:00:05
Core 1	2010.12.01 20:00:05	Open bar time of the chart symbol USDJPY: 2010.12.01 20:00
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 20:00:00	Waiting 5 seconds..
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 16:00:05	The bars are synchronized at 2010.12.01 16:00:05

Utilisez la fonction TimeTradeServer() au lieu de TimeCurrent(), si vous avez besoin d'obtenir l'heure actuelle du serveur, et non l'heure d'entrée du dernier tick.

Il y a une autre façon de synchroniser les barres - à l'aide du timer. L'exemple d'un tel EA Synchronize_Bars_Use_OnTimer.mq5 est attaché à cet article.

La fonction IndicatorRelease() dans le Strategy Tester

A la fin du test, le graphique de l'instrument s'ouvre automatiquement, et affiche les transactions réalisées et les indicateurs utilisés dans l'EA. Cela permet de vérifier visuellement les points d'entrée et de sortie, et de les comparer aux valeurs des indicateurs.

Remarque : les indicateurs affichés sur le graphique s'ouvrent automatiquement après l'achèvement du test, et sont calculés de nouveau après la fin du test. Même si ces indicateurs ont été utilisés dans l'EA testé.

Mais dans certains cas, le programmeur peut vouloir cacher les informations sur les indicateurs impliqués dans l'algorithme de trading. Par exemple, le code de l'EA est loué ou est vendu comme un fichier exécutable sans mise à disposition du code source. Pour cela, la fonction IndicatorRelease() est utile.

Si le terminal établit un modèle avec le nom tester.tpl dans le directory/profiles/templates du terminal client, alors il sera appliqué au graphique ouvert. En son absence, le modèle par défaut est appliqué (default.tpl).

© 2000-2024, MetaQuotes Ltd.

La fonction IndicatorRelease() est initialement prévue pour supprimer la partie du calcul de l'indicateur, si elle n'est plus nécessaire. Cela permet d'économiser la mémoire, ainsi que les ressources CPU, parce que chaque tick appelle un calcul de l'indicateur. Son deuxième objectif est d'interdire l'affichage d'un indicateur sur le graphique de test, après un seul essai.

Pour interdire l'affichage de l'indicateur sur le graphique à la fin du test, appelez IndicatorRelease() avec le handle de l'indicateur dans le gestionnaire OnDeinit(). La fonction OnDeinit() est toujours appelée après la fin et avant l'affichage du graphique du test.

```
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   bool hidden=IndicatorRelease(handle_ind);
   if(hidden) Print("IndicatorRelease() est exécuté avec succès ");
   else Print("IndicatorRelease() a retourné false. Code de l'erreur ",GetLastError())
  }
```

Pour interdire l'affichage de l'indicateur sur le graphique à la fin du test, utilisez la fonction IndicatorRelease() dans le gestionnaire OnDeinit().

Gestion des Evénements dans le Strategy Tester

La présence du gestionnaire OnTick() dans l'EA n'est pas obligatoire pour effectuer des tests sur les données historiques dans le MetaTrader 5 Tester. Il suffit que l'EA contienne au moins l'une des fonctions-gestionnaire suivantes :

- OnTick() - le gestionnaire de l'événement de l'arrivée d'un nouveau tick ;
- OnTrade() - le gestionnaire de l'événement de trading ;
- OnTimer() - le gestionnaire de l'événement de l'arrivée du signal du timer ;
- OnChartEvent() - le gestionnaire d'événements clients.

Lors du test dans un EA, nous pouvons gérer les événements personnalisés en utilisant la fonction OnChartEvent(), mais dans les indicateurs, cette fonction n'est pas appelée dans le testeur. Même si l'indicateur a le gestionnaire OnChartEvent() et que cet indicateur est utilisé dans l'EA testé, l'indicateur lui-même ne recevra pas les événements personnalisés.

Pendant le test, un indicateur peut générer des événements personnalisés en utilisant la fonction EventChartCustom(), et l'EA peut traiter cet événement dans la fonction OnChartEvent().

En plus de ces événements, des événements spéciaux associés aux processus de test et d'optimisation sont générés dans le Strategy Tester :

- Tester - cet événement est généré à la fin du test de l'EA sur les données historiques. L'événement Tester est géré en utilisant la fonction OnTester(). Cette fonction peut être utilisée seulement dans les EA de test et est destinée principalement au calcul d'une valeur qui est utilisée comme critère personnalisé maximum pour l'optimisation génétique des paramètres d'entrée.
- TesterInit - cet événement est généré au lancement de l'optimisation dans le Strategy Tester avant le premier passage. L'événement TesterInit est géré en utilisant la fonction OnTesterInit(). Au lancement de l'optimisation, un Expert Advisor ayant cette fonction est chargée automatiquement sur un graphique séparé du terminal avec le symbole et la période indiqués dans le testeur et reçoit

© 2000-2024, MetaQuotes Ltd.

l'évènement TesterInit. La fonction est utilisée pour initialiser un Expert Advisor avant le début de l'optimisation pour un traitement ultérieur des résultats de l'optimisation.

- TesterPass - cet événement est généré lorsqu'un nouveau bloc de données est reçu. L'événement TesterPass est géré en utilisant la fonction OnTesterPass(). Au lancement de l'optimisation, un Expert Advisor ayant cette fonction est chargée automatiquement sur un graphique séparé du terminal avec le symbole et la période indiqués dans le testeur et reçoit l'évènement TesterPass lorsqu'un bloc de données est reçu pendant l'optimisation. La fonction est utilisée pour le traitement dynamique des résultats d'optimisation "aussitôt", sans attendre son achèvement. Les frames sont ajoutés à l'aide de la fonction FrameAdd(), qui peut être appelé à la fin d'une passe unique dans le gestionnaire OnTester().
- TesterDeinit - cet événement est généré à l'issue de l'optimisation de l'Expert Advisor A dans le Strategy Tester. L'événement TesterDeinit est géré en utilisant la fonction OnTesterDeinit(). L'Expert Advisor qui a ce gestionnaire est chargé automatiquement sur un graphique au début de l'optimisation et reçoit l'événement TesterDeinit après son achèvement. La fonction est utilisée pour le traitement final de tous les résultats de l'optimisation.

Agents de Test

Les tests dans le terminal client MetaTrader 5 sont effectués en utilisant des agents de test. Les agents locaux sont créés et activés automatiquement. Le nombre d'agents locaux correspond par défaut au nombre de coeurs de l'ordinateur.

Chaque agent de test a sa propre copie des variables globales, qui ne sont pas liés au terminal client. Le terminal lui-même est le répartiteur qui distribue les tâches aux agents locaux et aux agents distants. Après l'exécution d'une tâche sur l'Expert Advisor avec les paramètres spécifiés, l'agent retourne les résultats au terminal. Pour un seul test, un seul agent est utilisé.

L'agent sauvegarde l'historique reçu du terminal dans des dossiers séparés, nommés avec le nom de l'instrument, de sorte que l'historique de la paire EURUSD est stocké dans un dossier nommé EURUSD. En outre, l'historique des instruments est séparé de leurs sources. La structure de stockage de l'historique est construit de la façon suivante :

```
répertoire_du testeur\Agent-AdresseIP-Port\bases\nom_de la source\history\nom_de l'ins
```

Par exemple, l'historique de la paire EURUSD du serveur MetaQuotes-Demo peut être stocké dans le dossier tester_catalog\Agent-127.0.0.1-3000\bases\MetaQuotes-Demo\EURUSD.

L'agent local passe en veille pendant 5 minutes après la fin du test, en attente de la prochaine tâche, afin de ne pas perdre de temps au lancement de l'appel suivant. L'agent local cesse son travail et est supprimé de la mémoire de l'ordinateur à l'expiration de l'attente.

Dans le cas d'une fin prématurée du test du côté de l'utilisateur (bouton "Annuler") ou en cas de fermeture du terminal client, tous les agents locaux arrêtent immédiatement leur travail et sont supprimés de la mémoire.

L'Echange des Données entre le Terminal et l'Agent

Lorsque vous exécutez un test, le terminal client se prépare à envoyer à un agent un certain nombre de blocs paramètres :

- Les paramètres d'entrée du test (le mode de simulation, l'intervalle du test, les instruments, le critère de l'optimisation, etc.)

© 2000-2024, MetaQuotes Ltd.

- La liste des symboles sélectionnés dans le "Market Watch"
- La spécification du symbole du test (la taille du contrat, les marges admissibles du marché pour définir un StopLoss et un TakeProfit, etc.)
- L'Expert Advisor testé et les valeurs de ses paramètres d'entrée
- Les informations sur les fichiers supplémentaires (les bibliothèques, les indicateurs, les fichiers de données - #property tester_...)

tester_indicator	string	Le nom de l'indicateur personnalisé au format "indicator_name.ex5". Les indicateurs qui nécessitent des tests sont définis automatiquement à partir de l'appel à la fonction iCustom() si le paramètre correspondant est défini par une chaîne de caractères constante. Cette propriété est requise pour tous les autres cas (utilisation de la fonctionIndicatorCreate() ou utilisation d'une chaîne de caractères non constante dans le paramètre qui définit le nom de l'indicateur)
tester_file	string	Le nom de fichier d'un testeur avec l'extension, entre guillemets (comme une chaîne de caractères constante). Le fichier spécifié sera transmis au testeur. Les fichiers d'entrée du test doivent être toujours spécifiés s'ils sont nécessaires.
tester_library	string	Le nom de la bibliothèque avec l'extension, entre guillemets. La bibliothèque peut avoir l'extension dll ou ex5. Les bibliothèques qui nécessitent des tests sont définies automatiquement. Toutefois, si l'une des bibliothèques est utilisé par un indicateur personnalisé, il est nécessaire d'utiliser la propriété donnée

Pour chaque bloc de paramètres, une empreinte numérique sous la forme de MD5-hash est créée, et envoyée à l'agent. Le MD5-hash est unique pour chaque ensemble, son volume est plus petit que le volume de l'information, cette propriété est requise.

Pour chaque bloc de paramètres, une empreinte digitale, sous la forme d'un hash MD5, est créée et envoyée à l'agent. L'agent reçoit un ensemble de blocs et les compare avec ceux qu'il a déjà. Si l'empreinte du bloc de paramètres donné n'est pas présent sur l'agent, ou si le hash reçu est différent de celui existant, l'agent demande ce bloc de paramètres. Cela permet de réduire le trafic entre le terminal et l'agent.

Après le test, l'agent retourne tous les résultats de la passe au terminal, qui sont ensuite affichés dans les onglets "Résultats du test" et "Résultats de l'optimisation" : le bénéfice réalisé, le nombre de transactions, le ratio de Sharpe, le résultat de la fonction OnTester(), etc.

Pendant l'optimisation, le terminal distribue les tâches aux agents par petits paquets, chaque paquet contenant plusieurs tâches (chaque tâche est un test simple avec un ensemble de paramètres d'entrée). Cela diminue le temps d'échange entre le terminal et l'agent.

Les agents n'enregistrent jamais les fichiers EX5 reçus du terminal sur le disque dur (l'Expert Advisor, les indicateurs, les bibliothèques, etc.) pour des raisons de sécurité, de sorte qu'un ordinateur avec un agent en cours d'exécution ne peut pas utiliser les données envoyées. Tous les autres fichiers, y compris les DLL, sont enregistrés dans la "sandbox". Vous ne pouvez pas tester les EA utilisant les DLL sur les agents distants.

© 2000-2024, MetaQuotes Ltd.

Les résultats des tests sont ajoutés au terminal dans un cache spécial des résultats (le cache des résultats) pour un accès rapide. Pour chaque ensemble de paramètres, le terminal recherche les résultats déjà disponibles à partir des tests précédents dans le cache des résultats afin d'éviter les redémarrages. Si le résultat d'un tel ensemble de paramètres n'a pas été trouvé, la tâche est donnée à l'agent pour effectuer le test.

Tout le trafic entre le terminal et l'agent est crypté.

Les ticks ne sont pas envoyés sur le réseau, ils sont générés sur les agents de test.

Utilisation du Dossier Partagés de Tous les Terminaux Client

Tout les agents de test sont isolés les uns des autres et du terminal client : chaque agent a un dossier personnel, dans lequel les journaux sont enregistrés. En outre toutes les opérations sur les fichiers pendant les tests de l'agent se produisent dans le dossier **nom_de l'agent/MQL5/Files**. Cependant, nous pouvons implémenter l'interaction entre les agents locaux et le terminal client à travers un dossier partagé pour tous les terminaux clients, si lors de l'ouverture du fichier vous spécifiez le drapeau FILE_COMMON :

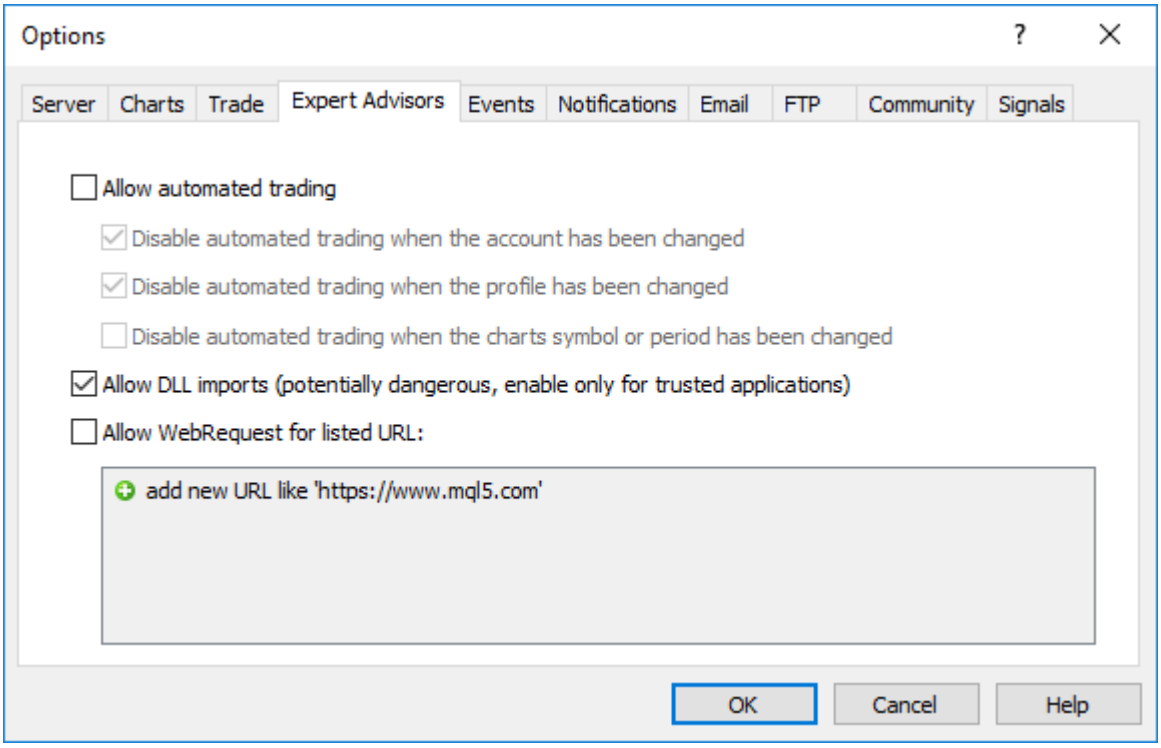
```
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le dossier partagé de tous les terminaux clients
   common_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- construction du nom de ce dossier
   PrintFormat("Ouvre le fichier dans le dossier partagé des terminaux client %s", cor
//--- ouvre le fichier dans le dossier partagé (indiqué par le drapeau FILE_COMMON)
   handle=FileOpen(filename,FILE_WRITE|FILE_READ|FILE_COMMON);
  ... d'autres actions
//---
   return(INIT_SUCCEEDED);
  }
```

Utilisation de DLL

Pour accélérer l'optimisation, nous pouvons utiliser non seulement les agents locaux, mais aussi les agents distants. Dans ce cas, il existe certaines limites pour les agents distants. Tout d'abord, les agents distants n'affichent pas dans leurs journaux les résultats de l'exécution de la fonction Print(), les messages sur l'ouverture et la fermeture des positions. Un minimum d'information est affichée dans le journal pour empêcher les EA mal écrits de remplir l'ordinateur sur lequel l'agent distant travaille avec des messages.

La deuxième limite est l'interdiction d'utiliser les DLL lors du test des EA. Les appels aux DLL sont absolument interdits sur les agents distants pour des raisons de sécurité. Sur les agents locaux, les appels aux DLL dans les EA testés sont autorisés uniquement avec l'autorisation correspondante "Autoriser l'importation de DLL".

© 2000-2024, MetaQuotes Ltd.



Remarque : lorsque vous utilisez des EA tiers (scripts, indicateurs) qui nécessitent des appels autorisés aux DLL, vous devez être conscient des risques que vous prenez lorsque vous autorisez cette option dans les paramètres du terminal. Indépendamment de comment l'EA sera utilisé - pour le test ou pour l'éxécution sur le graphique.

© 2000-2024, MetaQuotes Ltd.

Les variables prédéterminées

Pour chaque programme mql5 exécuté on soutient une série de variables prédéterminées, qui reflètent l'état du graphique courant de prix au moment du lancement du programme - l'expert, le script ou l'indicateur d'utilisateur.

Le terminal de client établit la valeur aux variables prédéterminées avant le lancement du programme mql5 à l'exécution. Les variables prédéterminées sont constantes et ne peuvent pas être changées du programme mql5. L'exception est faite pour la variable _LastError, qui peut être remis au zéro par la fonction ResetLastError.

Variable	Valeur
_AppliedTo	La variable _AppliedTo permet de trouver le type des données utilisées pour le calcul de l'indicateur
_Digits	Le nombre de signes décimaux après la virgule
_Point	La grandeur du point de l'instrument courant en devise de la cotation
_LastError	La signification de la dernière erreur
_Period	La signification du temps trame du graphique courant
_RandomSeed	L'état actuel du générateur des nombres entiers aléatoires
_StopFlag	Le drapeau de l'arrêt du programme
_Symbol	Le nom du symbole du graphique courant
_UninitReason	Le code de la raison de la déinitialisation du programme
_IsX64	La variable _IsX64 permet de trouver la version du terminal dans lequel l'application MQL5 est en cours d'exécution

Les bibliothèques se servent des variables du programme qui les a appelées.

© 2000-2024, MetaQuotes Ltd.

int _AppliedTo

La variable _AppliedTo permet de trouver le type des données utilisées pour le calcul de l'indicateur :

Type de Données	Signification	Description des données utilisées pour le calcul de l'indicateur.
—	0	L'indicateur utilise la 2ème forme de l'appel à OnCalculate() - les données pour le calcul ne sont pas spécifiées par un buffer ou par un tableau de données donné
Close	1	Prix de clôture
Open	2	Prix d'ouverture
High	3	Prix les plus hauts
Low	4	Prix les plus bas
Prix médians (HL/2)	5	Prix médians = (High+Low)/2
Prix typiques (HLC/3)	6	Prix typiques = (High+Low+Close)/3
Prix pondérés (HLCC/4)	7	Prix pondérés = (Open+High+Low+Close)/4
Données de l'Indicateur Précédent	8	Les données de l'indicateur qui a été placé sur le graphique juste avant cet indicateur
Données du Premier Indicateur	9	Les données de l'indicateur qui a été placé en premier sur le graphique
Handle de l'indicateur	10+	Les données de l'indicateur qui a été passé à la fonction iCustom() en utilisant le handle de l'indicateur. La valeur de _AppliedTo contient le handle de l'indicateur

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction d'initialisation d'un indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapping des buffers des indicateurs
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
// Récupère le type des données utilisées pour le calcul de l'indicateur
   Print("_AppliedTo=",_AppliedTo);
   Print(getIndicatorDataDescription(_AppliedTo));
```

© 2000-2024, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Description des données utilisées pour le calcul de l'indicateur |
//+------------------------------------------------------------------+
string getIndicatorDataDescription(int data_id)
  {
   string descr="";
   switch(data_id)
     {
      case(0):descr="C'est le 1er type de OnCalculate() - aucun buffer de données";
         break;
      case(1):descr="L'indicateur est calculé sur les prix Close";
         break;
      case(2):descr="L'indicateur est calculé sur les prix Open";
         break;
      case(3):descr="L'indicateur est calculé sur les prix High";
         break;
      case(4):descr="L'indicateur est calculé sur les prix Low";
         break;
      case(5):descr="L'indicateur est calculé sur les prix médians (HL/2)";
         break;
      case(6):descr="L'indicateur est calculé sur les prix typiques (HLC/3)";
         break;
      case(7):descr="L'indicateur est calculé sur les prix pondérés (HLCC/4)";
         break;
      case(8):descr="L'indicateur est calculé sur les Données de l'Indicateur Précéder
         break;
      case(9):descr="L'indicateur est calculé sur les Données du Premier Indicateur";
         break;
      default: descr="L'indicateur est calculé sur les données de l'indicateur avec le
         break;
     }
//---
   return descr;
  }
```

Voir aussi

ENUM_APPLIED_PRICE

© 2000-2024, MetaQuotes Ltd.

int _Digits

Dans la variable _Digits se trouve le nombre de signes décimaux après la virgule, qui définit l'exactitude de la mesure du prix du symbole du graphique courant.

On peut utiliser aussi la fonction Digits().

© 2000-2024, MetaQuotes Ltd.

double _Point

Dans la variable _Point se trouve la grandeur de point du symbole actuel dans la devise de la cotation.

On peut utiliser aussi la fonction Point().

© 2000-2024, MetaQuotes Ltd.

int _LastError

Dans la variable _LastError se trouve se trouve la signification de la dernière erreur, passé pendant l'exécution du programme mql5. On peut remettre la valeur au zéro par la fonction ResetLastError().

Pour obtenir le code de la dernière erreur, on peut utiliser aussi la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

ENUM_TIMEFRAMES _Period

Dans la variable _Period se trouve la signification du temps trame du graphique courant.

On peut utiliser aussi la fonction Period().

Voir aussi

PeriodSeconds, Les périodes des graphiques, La date et le temps, La visibilité des objets

© 2000-2024, MetaQuotes Ltd.

_RandomSeed

La variable pour stocker l'état actuel à la génération des nombres entiers aléatoires. _RandomSeedchange la valeur à l'appel de MathRand(). UtilisezMathSrand() pour définir l'état initial nécessaire.

Un nombre aléatoire **x**, reçu par la fonction MathRand(), est calculé à chaque appel comme ça:

```
x=_RandomSeed*214013+2531011;
_RandomSeed=x;
x=(x>>16)&0x7FFF;
```

Voir aussi

MathRand(), MathSrand(), Les types entiers

© 2000-2024, MetaQuotes Ltd.

bool _StopFlag

Dans la variable _StopFlag se trouve le drapeau de l'arrêt du programme mql5. Quand le terminal de client tente d'arrêter le programme, la valeur true s'inscrit à cette variable.

Pour vérifier l'état du drapeau _StopFlag on peut aussi utiliser la fonction IsStopped().

© 2000-2024, MetaQuotes Ltd.

string _Symbol

Dans la variable _Symbol se trouve le nom du symbole du graphique courant.

On peut utiliser aussi la fonction Symbol().

© 2000-2024, MetaQuotes Ltd.

int _UninitReason

Dans la variable _UninitReason se trouve le code de la raison de la déinitialisation du programme.

D'habitude on reçoit le code de la raison de la déinitialisation à l'aide de la fonction UninitializeReason().

© 2000-2024, MetaQuotes Ltd.

int _IsX64

La variable _IsX64 permet de trouver la version du terminal dans lequel l'application MQL5 est en cours d'exécution: _IsX64=0 pour un terminal 32 bits et _IsX64!=0 pour un terminal 64 bits.

La fonction TerminalInfoInteger(TERMINAL_X64) peut également être utilisée.

Exemple :

```
// Vérification du terminal dans lequel le programme tourne
   Print("_IsX64=",_IsX64);
   if(_IsX64)
      Print("Le programme ",__FILE__," s'exécute dans le terminal 64 bits");
   else
      Print("Le programme ",__FILE__," s'exécute dans le terminal 32");
   Print("TerminalInfoInteger(TERMINAL_X64)=",TerminalInfoInteger(TERMINAL_X64));
```

Voir aussi

MQLInfoInteger, Importer des fonctions (#import)

© 2000-2024, MetaQuotes Ltd.

Les fonctions communes

Les fonctions de la destination commune, qui ne sont pas entrées dans un des groupes spécialisés.

Fonction	Action
Alert	Affiche le message dans la fenêtre séparée
CheckPointer	Rend le type de l'indication de l'objet
Comment	Affiche le message à un angle gauche supérieur du graphique de prix
CryptEncode	Transforms the data from array with the specified method
CryptDecode	Performs the inverse transformation of the data from array, tranformed by CryptEncode()
DebugBreak	Le point d'arrêt de programme de débogage
ExpertRemove	Arrête le travail de l'expert et le décharge du graphique
GetPointer	Rend l'indicateur de l'objet
GetTickCount	Rend la quantité des millisecondes qui ont passé dès le moment du départ du système
GetTickCount64	Rend la quantité des millisecondes qui ont passé dès le moment du départ du système
GetMicrosecondCount	La fonction GetMicrosecondCount() retourne le nombre de microsecondes passées depuis le lancement du programme MQL5.
MessageBox	Crée, affiche la fenêtre des messages et la dirige
PeriodSeconds	Rend la quantité de secondes dans la période
PlaySound	Reproduit le fichier sonore
Print	Affiche un message dans le journal
PrintFormat	Formate et imprime l'ensemble des symboles et des valeurs dans un fichier-journal conformément au format donné
ResetLastError	Établit la valeur de la variable prédéterminée _LastError au zéro
ResourceCreate	Crée la ressource de l'image à la base de l'ensemble des données
ResourceFree	Supprime la ressource dynamiquement créée (libère la mémoire occupée par la ressource)
ResourceReadImage	Lit les données de la ressource graphique, créé par la fonction ResourceCreate() ou sauvegardée dans le fichier EX5 à la compilation
ResourceSave	Sauvegarde la ressource au fichier spécifié
SetUserError	Établit la variable prédéterminée _LastError à la valeur égale à ERR_USER_ERROR_FIRST + user_error

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
SetReturnError	Définit le code que le process du terminal retourne à la fin d'une opération
Sleep	Retient l'exécution de l'expert courant ou le script sur l'intervalle défini
TerminalClose	Envoie au terminal l'ordre sur l'achèvement du travail
TesterHideIndicators	Définit le mode d'affichage des indicateurs utilisés dans un EA.
TesterStatistics	Rend la valeur du paramètre indiqué statistique, calculée d'après les résultats du test
TesterStop	Envoie la commande de fin du programme lors d'un test
TesterDeposit	Fonction spéciale qui émule le dépôt de fonds pendant un test. Elle peut être utilisée dans certains système de money management
TesterWithdrawal	La fonction spéciale pour l'émulation des opérations du main levée des moyens pendant le test
TranslateKey	Retourne le caractère Unicode d'une touche de touche virtuel
ZeroMemory	Remet au zéro la variable transmise selon la référence. La variable peut être de n'importe quel type, seulement les classes et les structures qui ont les constructeurs font l'exception

© 2000-2024, MetaQuotes Ltd.

Alert

Affiche la fenêtre de dialogue, contenant les données d'utilisateur.

```
void  Alert(
   argument,      // première valeur
   ...            // valeurs ultérieures
   );
```

Paramètres

argument

[in] N'importe quelles valeurs divisées par les virgules. Pour la division de l'information déduite en quelques lignes on peut utiliser le symbole de la transmission de la ligne "\n" ou "\r\n". La quantité de paramètres ne peut pas excéder 64.

La valeur rendue

Il n'y a pas de valeur rendue

Note

On ne peut pas transmettre les tableaux à la fonction Alert(). Les tableaux doivent être déduits par les éléments. Les données de type double sont déduites avec 8 chiffres décimaux après le point, les données de type float sont déduites avec 5 chiffres décimaux après le point. Pour la sortie des nombres réels avec une autre exactitude ou dans le format scientifique il est nécessaire d'utiliser la fonction DoubleToString().

Les données du type bool sont déduites en forme des chaînes "true" ou "false". Les dates sont déduites comme YYYY.MM.DD HH:MI:SS. Pour la sortie de la date dans un autre format il est nécessaire d'utiliser la fonction TimeToString(). Les données du type color sont déduites en forme de la chaîne R,G,B, ou en forme du nom de la couleur, si cette couleur assiste dans l'ensemble des couleurs.

При работе в тестере стратегий функция Alert() не выполняется.

© 2000-2024, MetaQuotes Ltd.

CheckPointer

Rend le type de l'indicateur de l'objet.

```
ENUM_POINTER_TYPE  CheckPointer(
   object* anyobject      // indicateur de l'objet
   );
```

Paramètres

anyobject

[in] L'indicateur de l'objet.

La valeur rendue

Rend la valeur de l'énumération ENUM_POINTER_TYPE.

Note

La tentative de l'appel à l'indicateur incorrect amène à l'achèvement critique du programme. C'est pourquoi il y a une nécessité de l'utilisation de la fonction CheckPointer avant l'utilisation de l'indicateur. L'indicateur peut être incorrect dans les cas suivants:

- l'indicateur est égal à NULL;
- si l'objet était supprimé à l'aide de l'opérateur delete.

On peut utiliser cette fonction le contrôle de l'indicateur sur la validité. La valeur, distinguée du zéro, garantit qu'on peut avoir l'accès selon l'indicateur.

Pour valider rapidemen le pointeur, vous pouvez également utiliser l'opérateur "!" (exemple) qui vérifiera via un appel implicite à la fonction CheckPointer.

Exemple:

```
//+------------------------------------------------------------------+
//| Supprimer la liste en suppriment ses éléments                    |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- indicateur de service pour le travail dans la boucle
   CItem* item;
//--- passons par la boucle et essayez de supprimer les indicateurs dynamiques
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Dynamic object ",item.Identifier()," to be deleted");
         delete (item);
        }
      else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
     }
//---
  }
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Les indicateurs des objets, Le contrôle de l'indicateur de l'objet, L'opérateur de l'effacement de l'objet delete

© 2000-2024, MetaQuotes Ltd.

Comment

Déduit le commentaire défini par l'utilisateur, à un angle gauche supérieur du graphique.

```
void  Comment(
   argument,      // première valeur
   ...            // valeurs ultérieures
   );
```

Paramètres

...

[in] N'importe quelles valeurs divisées par les virgules. Pour placer l'information sur plusieurs lignes, on peut utiliser le symbole du retour chariot "\n" ou "\r\n". Le nombre de paramètres ne peut pas excéder 64. La longueur totale du message (y compris les symboles invisibles) ne peut pas excéder 2045 symboles (les symboles superflus seront coupés à la sortie.

Valeur de Retour

Aucune

Note

On ne peut pas transmettre de tableaux à la fonction Comment(). Les tableaux doivent être imprimés par éléments.

Les données du type double sont écrites à 16 chiffres décimaux après le point. De plus, les données peuvent être écrites au format traditionnel ou scientifique - suivant le format le plus compact. Les données du type float sont écrites avec 5 chiffres décimaux après le point. Pour l'affichage des nombres réels avec une autre exactitude ou dans un format spécifié, il est nécessaire d'utiliser la fonction DoubleToString().

Les données du type bool sont écrites sous la forme des chaînes de caractères "true" et "false". Les dates sont écrites au format YYYY.MM.DD HH:MI:SS. Pour les dates dans un autre format, il est nécessaire d'utiliser la fonction TimeToString(). Les données du type color sont écrites au format R,G,B, ou avec le nom de la couleur, si cette couleur existe dans l'ensemble des couleurs.

La fonction Comment() ne fonctione pas pendant l'optimisation dans le Strategy Tester.

Exemple :

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
//--- Ecrit les valeurs sur trois lignes
   Comment(StringFormat("Prix\nAsk = %G\nBid = %G\nSpread = %d",Ask,Bid,Spread));
  }
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

ChartSetString, ChartGetString

© 2000-2024, MetaQuotes Ltd.

CryptEncode

Transforms the data from array with the specified method.

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,         // method
   const uchar&        data[],         // source array
   const uchar&        key[],          // key
   uchar&              result[]        // destination array
   );
```

Parameters

method

[in] Data transformation method. Can be one of the values of ENUM_CRYPT_METHOD enumeration.

data[]

[in] Source array.

key[]

[in] Key array.

result[]

[out] Destination array.

Return Value

Amount of bytes in the destination array or 0 in case of error. To obtain information about the error call the GetLastError() function.

Example:

```
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";
//--- check
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);
//--- transform to HEX string
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   string text="The quick brown fox jumps over the lazy dog";
   string keystr="ABCDEFG";
   uchar src[],dst[],key[];
//--- prepare key
   StringToCharArray(keystr,key);
//--- copy text to source array src[]
   StringToCharArray(text,src);
//--- print initial data
   PrintFormat("Initial data: size=%d, string='%s'",ArraySize(src),CharArrayToString(s
//--- encrypt src[] with DES 56-bit key in key[]
   int res=CryptEncode(CRYPT_DES,src,key,dst);
//--- check error
   if(res>0)
     {
      //--- print encrypted data
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
      //--- decode dst[] to src[]
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- check error
      if(res>0)
        {
         //--- print decoded data
         PrintFormat("Decoded data: size=%d, string='%s'",ArraySize(src),CharArrayToSt
        }
      else
         Print("Error in CryptDecode. Error code=",GetLastError());
     }
   else
      Print("Error in CryptEncode. Error code=",GetLastError());
  }
```

See also

Array Functions, CryptDecode()

© 2000-2024, MetaQuotes Ltd.

CryptDecode

Performs the inverse transformation of the data from array, tranformed by CryptEncode().

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,         // method
   const uchar&        data[],         // source array
   const uchar&        key[],          // key
   uchar&              result[]        // destination array
   );
```

Parameters

method

[in] Data transformation method. Can be one of the values of ENUM_CRYPT_METHOD enumeration.

data[]

[in] Source array.

key[]

[in] Key array.

result[]

[out] Destination array.

Return Value

Amount of bytes in the destination array or 0 in case of error. To obtain information about the error call the GetLastError() function.

See also

Array Functions, CryptEncode()

© 2000-2024, MetaQuotes Ltd.

DebugBreak

Le point d'arrêt de programme de débogage.

```
void  DebugBreak();
```

La valeur rendue

Il n'y a pas de valeur rendue.

Note

L'interruption de l'exécution du programme mql5 se passe seulement dans le cas où le programme est lancé en régime du réglage. On peut utiliser la fonction pour l'affichage des valeurs des variables et/ou une exécution ultérieure pas à pas.

© 2000-2024, MetaQuotes Ltd.

ExpertRemove

Arrête le travail de l'expert et la décharge du graphique.

```
void  ExpertRemove();
```

La valeur rendue

Il n'y a pas de valeur rendue.

Note

L'arrêt de l'expert ne se passe pas immédiatement à l'appel de la fonction ExpertRemove(), on produit seulement la section du drapeau pour l'arrêt du travail de l'expert. C'est-à-dire, l'expert ne traitera pas déjà un événement suivant, on produira l'appel OnDeinit() et l'effacement avec l'éloignement du graphique.

Appeler ExpertRemove() dans le strategy tester dans la fonction OnInit() annule le test de l'ensemble de paramètres courant. Une telle fin est considérée comme étant une erreur d'initialisation.

Lors de l'appel à ExpertRemove() dans le strategy tester après l'initialisation réussie d'un EA, un test est effectué normalement avec l'appel à OnDeinit() et OnTester(). Dans ce cas, les statistiques complètes de trading et la valeur du critère d'optimisation sont obtenues.

Exemple:

```
//+------------------------------------------------------------------+
//|                                          Test_ExpertRemove.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
input int ticks_to_close=20;// nombre de ticks avant la décharge de l'expert
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(TimeCurrent(),": " ,__FUNCTION__," reason code = ",reason);
//--- "clear" comment
   Comment("");
//---
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   static int tick_counter=0;
//---
   tick_counter++;
   Comment("\nAvant la décharge de l'expert ",__FILE__," reste ",
           (ticks_to_close-tick_counter)," des ticks");
//--- jusqu'au
```

© 2000-2024, MetaQuotes Ltd.

```
    if(tick_counter>=ticks_to_close)
      {
       ExpertRemove();
       Print(TimeCurrent(),": ",__FUNCTION__," expert sera déchargé ");
      }
    Print("tick_counter = ",tick_counter);
//---
  }
//+------------------------------------------------------------------+
```

Voir aussi

L'exécution des programmes, Les événements du terminal de client

© 2000-2024, MetaQuotes Ltd.

GetPointer

Rend l'indicateur de l'objet.

```
void*  GetPointer(
   any_class anyobject      // objet de n'importe quelle classe
   );
```

Paramètres

anyobject

[in] L'objet de n'importe quelle classe.

La valeur rendue

Rend l'indicateur de l'objet.

Note

Seulement les objets des classes ont les indicateurs. Les exemplaires des structures et les variables des types simples des indicateurs n'ont pas des indicateurs. L'objet de la classe, qui n'est pas créé à l'aide de l'opérateur new(), mais par exemple, automatiquement créé dans le tableau des objets, a l'indicateur tout de même. Seulement cet indicateur aura le type automatique POINTER_AUTOMATIC,et on ne peut pas lui appliquer l'opérateur delete(). Pour le reste l'indicateur du type ne se distingue pas des indicateurs dynamiques du type POINTER_DYNAMIC.

Puisque les variables comme les structures et les types simples n'ont pas les indicateurs, il est interdit d'appliquer la fonction GetPointer() à eux. Il est aussi interdit de transmettre l'indicateur comme l'argument de la fonction. Dans tous les cas énumérés le compilateur notifiera d'une erreur.

La tentative de l'appel à l'indicateur incorrect amène à l'achèvement critique du programme. C'est pourquoi il y a une nécessité de l'utilisation de la fonction CheckPointer() avant l'utilisation de l'indicateur. L'indicateur peut être incorrect dans les cas suivants:

- l'indicateur est égal à NULL;
- si l'objet était supprimé à l'aide de l'opérateur delete.

On peut utiliser cette fonction le contrôle de l'indicateur sur la validité. La valeur, distinguée du zéro, garantit qu'on peut avoir l'accès selon l'indicateur.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              Check_GetPointer.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

//+------------------------------------------------------------------+
//| Classe en exécutant l'élément de liste                           |
//+------------------------------------------------------------------+
class CItem
  {
   int               m_id;
   string            m_comment;
   CItem*            m_next;
public:
                     CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
                    ~CItem() { Print("Destructor of ",m_id,
                                     (CheckPointer(GetPointer(this))==POINTER_DYNAMIC)
                                     "dynamic":"non-dynamic"); }
   void              Initialize(int id,string comm) { m_id=id; m_comment=comm; }
   void              PrintMe() { Print(__FUNCTION__,":",m_id,m_comment); }
   int               Identifier() { return(m_id); }
   CItem*            Next() {return(m_next); }
   void              Next(CItem *item) { m_next=item; }
  };
//+------------------------------------------------------------------+
//| Classe la plus simple de la liste                                |
//+------------------------------------------------------------------+
class CMyList
  {
   CItem*            m_items;
public:
                     CMyList() { m_items=NULL; }
                    ~CMyList() { Destroy(); }
    bool             InsertToBegin(CItem* item);
    void             Destroy();
  };
//+------------------------------------------------------------------+
//| Insérer l'élément de liste au commencement                       |
//+------------------------------------------------------------------+
bool CMyList::InsertToBegin(CItem* item)
  {
   if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
   item.Next(m_items);
   m_items=item;
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprimer la liste en supprimant des éléments                    |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- indicateur de service pour travailler dans une boucle
   CItem* item;
//--- passons par la boucle et essayez de supprimer les indicateurs dynamiques
   while(CheckPointer(m_items)!=POINTER_INVALID)
```

```
      {
       item=m_items;
       m_items=m_items.Next();
       if(CheckPointer(item)==POINTER_DYNAMIC)
         {
          Print("Dynamyc object ",item.Identifier()," to be deleted");
          delete (item);
         }
       else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
      }
//---
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   CMyList list;
   CItem   items[10];
   CItem*  item;
//--- créons et ajouteront à la liste l'indicateur dynamique de l'objet
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(100,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- ajoutons les indicateurs automatiques à la liste
   for(int i=0; i<10; i++)
     {
      items[i].Initialize(i,"automatic");
      items[i].PrintMe();
      item=GetPointer(items[i]);
      if(CheckPointer(item)!=POINTER_INVALID)
         list.InsertToBegin(item);
     }
//--- ajoutons encore un indicateur dynamique de l'objet au début de la liste
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(200,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//---  supprimons les éléments de la liste
   list.Destroy();
//---  tous les éléments de la liste seront supprimés,
//--- regarde le signet Experts dans le terminal
  }
```

Voir aussi

Les indicateurs des objets, Le contrôle de l'indicateur de l'objet , L'opérateur de l'effacement de l'objet delete

© 2000-2024, MetaQuotes Ltd.

GetTickCount

La fonction GetTickCount() rend la quantité des millisecondes qui ont passé depuis le début de système.

```
uint  GetTickCount();
```

La valeur rendue

La valeur du type uint.

Note

Le compteur est limité par la résolution systémique de minuteur. Puisque le temps se stocke comme l'entier sans signes, il déborde tous les 49.7 jours au travail continu de l'ordinateur.

Exemple:

```
#define MAX_SIZE 40
//+------------------------------------------------------------------+
/ / | le script pour mesurer le temps de calcul de 40 nombres de Fibonacci |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- rappelons la valeur initiale
   uint start=GetTickCount();
//--- la variable pour la réception du numéro suivant dans la suite de Fibonacci
   long fib=0;
//--- la boucle dans laquelle on calcule un certain nombre de numéros dans la suite de
   for(int i=0;i<MAX_SIZE;i++) fib=TestFibo(i);
//--- recevrons le temps dépensé en millisecondes
   uint time=GetTickCount()-start;
//--- déduisons au journal "Experts" le message
   PrintFormat("Le calcul %d des premiers nombres Fibonacci a pris %d ms",MAX_SIZE,tir
//--- le travail du script est terminé
   return;
  }
//+------------------------------------------------------------------+
//| La fonction de la réception du nombre de Fibonacci selon son numéro d'ordre    |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- le premier membre de la suite Fibonacci
   if(n<2) return(1);
//--- tous les membres suivants sont calculés par cette formule
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

Voir aussi

Date et heure, EventSetMillisecondTimer, GetTickCount64, GetMicrosecondCount

© 2000-2024, MetaQuotes Ltd.

GetTickCount64

La fonction GetTickCount64() retourne le nombre de millisecondes depuis le dernier lancement du système.

```
ulong  GetTickCount64();
```

Valeur de Retour

Une valeur de type ulong.

Note

Le compteur est limité à la précision du timer du système, qui retourne habituellement un résultat avec une précision entre 10 et 16 millisecondes. Contrairement à GetTickCount, qui est de type uint et dont le contenu déborde tous les 49,7 jours dans le cas d'une utilisation continue de l'ordinateur, la fonction GetTickCount64() peut être utilisée pour indéfiniment et n'est pas sujette au débordement (overflow).

Voir aussi

Date et Heure, EventSetMillisecondTimer, GetTickCount, GetMicrosecondCount

© 2000-2024, MetaQuotes Ltd.

GetMicrosecondCount

La fonction GetMicrosecondCount() retourne le nombre de microsecondes passées depuis le lancement du programme MQL5.

```
ulong  GetMicrosecondCount();
```

Valeur de Retour

Valeur de type ulong.

Exemple:

```
//+------------------------------------------------------------------+
//| Fonction de test                                                 |
//+------------------------------------------------------------------+
void Test()
  {
   int    res_int=0;
   double res_double=0;
//---
   for(int i=0;i<10000;i++)
     {
      res_int+=i*i;
      res_int++;
      res_double+=i*i;
      res_double++;
     }
  }
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
   uint   ui=0,ui_max=0,ui_min=INT_MAX;
   ulong  ul=0,ul_max=0,ul_min=INT_MAX;
//--- nombre de mesures
   for(int count=0;count<1000;count++)
     {
      uint  ui_res=0;
      ulong ul_res=0;
      //---
      for(int n=0;n<2;n++)
        {
         //---  sélction du type de mesure
         if(n==0) ui=GetTickCount();
         else     ul=GetMicrosecondCount();
         //--- exécute le code
         Test();
         //--- ajoute le résultat de la mesure (suivant le type)
```

© 2000-2024, MetaQuotes Ltd.

```
          if(n==0) ui_res+=GetTickCount()-ui;
          else     ul_res+=GetMicrosecondCount()-ul;
        }
      //--- calcul les heures min et max pour les 2 mesures
      if(ui_min>ui_res) ui_min=ui_res;
      if(ui_max<ui_res) ui_max=ui_res;
      if(ul_min>ul_res) ul_min=ul_res;
      if(ul_max<ul_res) ul_max=ul_res;
     }
//---
   Print("Erreur GetTickCount (msec): ",ui_max-ui_min);
   Print("Erreur GetMicrosecondCount (msec): ",DoubleToString((ul_max-ul_min)/1000.0,2
  }
```

Voir aussi

Date et Heure, GetTickCount, GetTickCount64

© 2000-2024, MetaQuotes Ltd.

MessageBox

Crée et affiche la fenêtre des messages et le dirige. La fenêtre des messages contient le message et le titre, n'importe quelle combinaison des signes prédéterminés et les boutons de commande.

```
int  MessageBox(
   string  text,              // texte de message
   string  caption=NULL,      // titre de la fenêtre
   int     flags=0            // définit l'ensemble des boutons dans la fenêtre
   );
```

Paramètres

text

[in] Le texte contenant le message pour l'affichage.

caption=NULL

[in] Le texte non obligatoire pour l'affichage dans le titre de la fenêtre du message. Si ce paramètre vide, le nom de l'expert sera affichée dans le titre de la fenêtre.

flags=0

[in] Les drapeaux non obligatoires définissant l'aspect et le comportement de la fenêtre de dialogue. Les drapeaux peuvent être la combinaison du groupe spécial des drapeaux.

La valeur rendue

Si la fonction est accomplie avec succès, la valeur rendue - une des valeurs des codes du retour MessageBox().

Note

La fonction ne peut pas être utilisée dans les indicateurs personnalisés car l'appel de MessageBox() suspend le thread d'exécution pendant tout le temps en attendant la réponse de l'utilisateur. Étant donné que tous les indicateurs pour chaque symbole sont exécutés dans un seul thread, une telle suspension rend le fonctionnement de tous les graphiques sur toutes les périodes pour ce symbole impossible.

La fonction MessageBox () ne fonctionne pas dans Strategy Tester.

© 2000-2024, MetaQuotes Ltd.

PeriodSeconds

Rend la quantité de secondes dans la période.

```
int  PeriodSeconds(
   ENUM_TIMEFRAMES  period=PERIOD_CURRENT      // période du graphique
   );
```

Paramètres

period=PERIOD_CURRENT

[in] La valeur de la période du graphique de l'énumération ENUM_TIMEFRAMES. Si le paramètre n'est pas spécifié, il rend le nombre de secondes de la période de graphique actuelle, à laquelle le programme est lancé.

La valeur rendue

La quantité de secondes dans la période indiquée.

Voir aussi

_Period, Les périodes des graphiques, La date et le temps, La visibilité des objets

© 2000-2024, MetaQuotes Ltd.

PlaySound

Reproduit le fichier sonore.

```
bool  PlaySound(
   string  filename      // nom du fichier
   );
```

Paramètres

filename

[in] Le chemin vers le fichier sonore. Если filename=NULL, воспроизведение звука прекращается.

La valeur rendue

true - si le fichier sonore est trouvé, autrement rend false.

Note

Le fichier doit être disposé dans le répertoire le répertoire_du terminal\Sounds ou son sous-catalogue. On reproduit seulement les fichiers sonores dans le format WAV.

Вызов PlaySound() с параметром NULL останавливает воспроизведение звука.

При работе в тестере стратегий функция PlaySound() не выполняется.

Voir aussi

Ressources

© 2000-2024, MetaQuotes Ltd.

Print

Imprime un certain message au journal des experts. Les paramètres peuvent avoir n'importe quel type.

```
void  Print(
   argument,      // première valeur
   ...            // valeurs ultérieures
   );
```

Paramètres

...

[in] N'importe quelles valeurs divisées par les virgules.La quantité de paramètres ne peut pas excéder 64.

Note

On ne peut pas transmettre les tableaux à la fonction Print(). Les tableaux doivent être déduits par les éléments.

Les données de type double sont déduites à près de 16 chiffres décimaux après le point, de plus les données peuvent être déduites au format traditionnel ou dans le format scientifique - suivant quelle inscription sera plus compacte. Les données du type float sont déduites avec 5 chiffres décimaux après le point. Pour la sortie des nombres réels avec une autre exactitude ou dans le format spécifié il est nécessaire d'utiliser la fonction PrintFormat().

Les données du type bool sont déduites en forme des chaînes "true" et "false". Les dates sont déduites comme YYYY.MM.DD HH:MI:SS. Pour la sortie de la date dans un autre format il est nécessaire d'utiliser la fonction TimeToString(). Les données du type color sont déduites en forme de la chaîne R,G,B, ou en forme du nom de la couleur, si cette couleur assiste dans l'ensemble des couleurs.

При работе в тестере стратегий в режиме оптимизации функция Print() не выполняется.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- déduisons DBL_MAX à l'aide de Print(), c'est égal à PrintFormat(%%.16G,DBL_MAX)
   Print("---- comment apparaît DBL_MAX -----");
   Print("Print(DBL_MAX)=",DBL_MAX);
//---maintenant nous déduisons le nombre DBL_MAX à l'aide de PrintFormat()
   PrintFormat("PrintFormat(%%.16G,DBL_MAX)=%.16G",DBL_MAX);
//--- Déduit au journal "Experts"
// Print(DBL_MAX)=1.797693134862316e+308
// PrintFormat(%.16G,DBL_MAX)=1.797693134862316E+308

//--- regardons comment on déduit le type float
   float c=(float)M_PI; // il faut évidemment rendre égal au type cible
   Print("c=",c, "    Pi=",M_PI, "    (float)M_PI=",(float)M_PI);
// c=3.14159    Pi=3.141592653589793    (float)M_PI=3.14159

//--- montrons ce qui peut se passer aux opérations arithmétiques sur les types réels
   double a=7,b=200;
   Print("---- avant les opérations arithmétiques");
   Print("a=",a,"   b=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- devisons a sur b (7/200)
   a=a/b;
//---comme si nous restaurons la valeur dans la variable b maintenant
   b=7.0/a; // il est prévu que b=7.0/(7.0/200.0)=>7.0/7.0*200.0=200 - mais ce n'est p
//--- déduisons une nouvelle valeur calculée b
   Print("-----après les opérations arithmétiques");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- déduit au journal "Experts"
// Print(b)=200.0
// Print(DoubleToString(b,16))=199.9999999999999716 ( voyons qu'en fait b n'est pas éq

//--- créons une très petite valeur epsilon=1E-013
   double epsilon=1e-13;
   Print("---- créons un très petit nombre");
   Print("epsilon=",epsilon); // recevrons  epsilon=1E-013
//--- maintenant nous décomptons l'epsilon du nombre b et nous déduisons de nouveau la
   b=b-epsilon;
//--- déduisons par deux moyens
   Print("---- après la soustraction d'epsilon de la variable b");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- déduit au journal "Experts"
// Print(b)=199.9999999999999  (maintenant la valeur b après la soustraction de l'epsi
// Print(DoubleToString(b,16))=199.9999999999998578
//    (maintenant la valeur b après la soustraction de l'epsilon ne peut pas être arro
  }
```

Voir aussi

DoubleToString, StringFormat

© 2000-2024, MetaQuotes Ltd.

PrintFormat

Formate et imprime l'ensemble des symboles et des valeurs dans un fichier-journal conformément au format donné.

```
void  PrintFormat(
   string format_string,    // ligne de format
   ...                      // valeurs des types simples
   );
```

Paramètres

format_string

[in] La ligne du format comprend les symboles ordinaires et encore et les spécifications du format, si la ligne de format est suivie par les arguments.

...

[in] N'importe quelles valeurs divisées par les virgules.La quantité de paramètres ne peut pas excéder 64, y compris la ligne de format.

La valeur rendue

La chaîne.

Note

При работе в тестере стратегий в режиме оптимизации функция PrintFormat() не выполняется.

Le nombre, l'ordre et le type de paramètres doit correspondre exactement à la composition des spécificateurs, sinon le résultat de l'impression est indéfini. Au lieu de la fonction PrintFormat() on peut utiliser la fonction printf().

Si la ligne du format est suivie encore par les paramètres, cette ligne doit contenir les spécifications du format définissant le format de la sortie de ces paramètres. La spécification du format commence toujours par le symbole du signe du pour-cent (%).

La ligne du format est lue de gauche à droite. Quand il y a une première spécification du format (si elle est), la valeur du premier paramètre après la ligne du format est transformée et est déduit selon la spécification donnée. La deuxième spécification du format appelle la transformation et la conclusion du deuxième paramètre etc, jusqu'à la fin de la ligne du format.

La spécification du format a la forme suivante:

%[flags][width][.precision][{h | l | ll | I32 | I64}]type

Chaque champ de la spécification de format est ou le symbole simple, ou le nombre désignant l'option ordinaire de format. La spécification la plus simple du format contient seulement le signe du pour-cent (%) et le symbole définissant le type du paramètre déduit (par exemple %s). S'il faut déduire dans la ligne de format le symbole le signe du pour-cent, il est nécessaire d'utiliser la spécification de format %%.

flags

© 2000-2024, MetaQuotes Ltd.

Drapeau	Description	Comportement par défaut
- (moins)	L'alignement selon le bout gauche dans la limite de la largeur donnée	L'alignement selon le bord droit
+ (plus)	La sortie du signe + ou - pour les valeurs des types des signes	Le signe est déduit seulement si la valeur est négative
0 (zéro)	Devant la valeur déduite on ajoute les zéros dans la limite de la largeur donnée. Si on a indiqué le drapeau **0** avec le format entier (**i**, **u**, **x**, **X**, **o**, **d**) et la spécification de l'exactitude est spécifiée (par exemple, %04.d), 0 est ignoré.	Rien n'est pas inséré
espace	Devant la valeur déduite on met l'espace, si la valeur est de signe et positive	Les espaces ne sont pas insérés
#	S'il est utilisé en commun avec le format **o**, **x** ou **X**, devant la valeur déduite est ajouté 0, 0x ou 0X conformément.	Rien n'est pas inséré
	S'il est utilisé en commun avec le format **e**, **E**, **a** ou **A**, cette valeur est déduite toujours avec le point décimal.	Le point décimal est déduit seulement s'il y a une partie non zéro fractionnaire.
	S'il est utilisé en commun avec le format **g** ou **G**, le drapeau définit la présence du point décimal dans la valeur déduite et empêche l'interruption des zéros principaux Le drapeau # est ignoré à l'utilisation commune avec les formats **c**, **d**, **i**, **u**, **s**.	Le point décimal est déduit seulement s'il y a une partie non zéro fractionnaire. Les zéros principaux sont coupés

width

Le nombre non négatif décimal, qui spécifie le nombre minimal des symboles déduits de la valeur formaté. Si la quantité de symboles déduits est moins de la largeur indiquée, on ajoute la quantité correspondante des espaces à gauche ou à droite suivant l'alignement (le drapeau -). En présence du drapeau le zéro (0), devant la valeur déduite est ajouté la quantité correspondante de zéros. Si le nombre de symboles déduits plus de largeur spécifiée, la valeur déduite n'est jamais tronquée

Si à titre de la largeur on indique l'astérisque (*), dans la liste des paramètres transmis à la place correspondante doit être une valeur du type int, qui sera utilisé pour l'indication de la largeur de la valeur déduite.

precision

Le nombre non négatif décimal, qui définit l'exactitude de lasortie - la quantité de chiffres après le point décimal. À la différence de la spécification de la largeur, la spécification de l'exactitude peut couper la partie de la signification fractionnaire avec l'arrondissement ou sans arrondissement

Pour des différents types (**type**)de format la spécification de l'exactitude est appliquée différemment.

© 2000-2024, MetaQuotes Ltd.

Types	Description	Comportement par défaut
a, A	La spécification de l'exactitude indique la quantité de signes après le point décimal	L'exactitude par défaut - 6.
c, C	N'est pas appliqué	
d, i, u, o, x, X	Indique le nombre minimal des chiffres déduits. Si le nombre de chiffres dans un paramètre correspondant est moins que exactitude indiquée les zéros sont ajoutés à la valeur déduite. La valeur déduite ne se coupe pas, si la quantité de chiffres déduits plus d'exactitude indiquée	L'exactitude par défaut - 1.
e, E, f	On indique le nombre de chiffres déduits après le point décimal. Le dernier chiffre déduit s'arrondit	L'exactitude par défaut - 6. Si on indique l'exactitude 0 ou la partie fractionnaire manque, le point décimal n'est pas déduit.
g, G	On indique le nombre maximum des chiffres signifiants	On déduit 6 chiffres signifiants.
s	On indique la quantité de symboles déduits de la ligne. Si la longueur la ligne excède la valeur de l'exactitude, la ligne est tronquée sur la sortie	On déduit toute la ligne

```
PrintFormat("1. %s", _Symbol);
PrintFormat("2. %.3s", _Symbol);
int length=4;
PrintFormat("3. %.*s", length, _Symbol);
/*
1. EURUSD
2. EUR
3. EURU
/
```

h | l | ll | I32 | I64

Les spécifications des grandeurs des données transmises à titre du paramètre.

Type du para-mètre	Préfixe utilisé	Spécificateur commun du type
int	l (minuscule L)	d, i, o, x, or X
uint	l (minuscule L)	o, u, x, or X

© 2000-2024, MetaQuotes Ltd.

Type du para-mètre	Préfixe utilisé	Spécificateur commun du type
long	ll (deux minuscules L)	d, i, o, x, or X
short	h	d, i, o, x, or X
ushort	h	o, u, x, or X
int	I32	d, i, o, x, or X
uint	I32	o, u, x, or X
long	I64	d, i, o, x, or X
ulong	I64	o, u, x, or X

type

Le spécificateur du type est le seul champ obligatoire pour la sortie formatée.

Symbole	Type	Format de sortie
c	int	Le symbole du type short (Unicode)
C	int	Le symbole du type char (ANSI)
d	int	L'entier de signe décimal
i	int	L'entier de signe décimal
o	int	Le nombre entier octal sans signe
u	int	Le nombre entier décimal sans signe
x	int	Le nombre entier hexadecimal sans signe avec l'utilisation "abcdef"
X	int	Le nombre entier hexadecimal sans signe avec l'utilisation "ABCDEF"
e	double	La valeur réelle au format [-]d.dddd **e** [sign]ddd, où d - un chiffre décimal, dddd - un ou plus des chiffres décimaux, ddd -le nombre à trois chiffres, qui détermine la grandeur de l'exposant, sign - le signe plus ou le moins
E	double	Identiquement au format e, excepté que le signe d'exposant est déduit par la majuscule (**E** au lieu d'e)
f	double	La valeur réelle au format [-]dddd.dddd, où dddd - un ou plus de chiffres décimaux. La quantité de signes déduits devant le point décimal dépend de la valeur de la signification du nombre. Le nombre de signes après le point décimal dépend de l'exactitude demandée.

© 2000-2024, MetaQuotes Ltd.

Symbole	Type	Format de sortie
g	double	La valeur réelle, déduite au format **f** ou **e**, en fonction de celui-là quelle sortie sera plus compacte.
G	double	La valeur réelle, déduite au format **f** ou **E**, en fonction de celui-là quelle sortie sera plus compacte.
a	double	La valeur réelle au format [−]0xh.hhhh **p**±dd, où h.hhhh - la mantisse dans la forme de chiffres hexadécimaux avec l'utilisation"abcdef", dd - un ou plus de chiffres de l'exposant. Le nombre de signes après la virgule est définie par la spécification de l'exactitude
A	double	La valeur réelle au format [−]0xh.hhhh **P**±dd, où h.hhhh - la mantisse dans la forme de chiffres hexadécimaux avec l'utilisation "ABCDEF", dd - un ou plus de chiffres de l'exposant. Le nombre de signes après la virgule est définie par la spécification de l'exactitude
s	string	La sortie de la ligne

Au lieu de la fonction PrintFormat() on peut utiliser la fonction printf().

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- имя торгового сервера
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- номер торговго счета
   int login=(int)AccountInfoInteger(ACCOUNT_LOGIN);
//--- вывод значения long
   long leverage=AccountInfoInteger(ACCOUNT_LEVERAGE);
   PrintFormat("%s %d: плечо = 1:%I64d",
               server,login,leverage);
//--- валюта депозита
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- вывод значения double с 2 цифрами после десятичной точки
   double equity=AccountInfoDouble(ACCOUNT_EQUITY);
   PrintFormat("%s %d: размер собственных средств на счете = %.2f %s",
               server,login,equity,currency);
//--- вывод значения double с обязательным выводом знака +/-
   double profit=AccountInfoDouble(ACCOUNT_PROFIT);
   PrintFormat("%s %d: текущий результат по открытым позициям = %+.2f %s",
               server,login,profit,currency);
//--- вывод значения double с переменным количеством цифр после десятичной точки
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   string format_string=StringFormat("%%s: значение одного пункта  = %%.%df",_Digits);
   PrintFormat(format_string,_Symbol,point_value);
//--- вывод значения int
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   PrintFormat("%s: текущий спред в пунктах = %d ",
               _Symbol,spread);
//--- вывод значения double в научном формате с плавающей запятой и точностью 17 знача
   PrintFormat("DBL_MAX = %.17e",DBL_MAX);
//--- вывод значения double в научном формате с плавающей запятой и точностью 17 знача
   PrintFormat("EMPTY_VALUE = %.17e",EMPTY_VALUE);
//--- вывод через PrintFormat() с точностью по умолчанию
   PrintFormat("PrintFormat(EMPTY_VALUE) = %e",EMPTY_VALUE);
//--- простой вывод через Print()
   Print("Print(EMPTY_VALUE) = ",EMPTY_VALUE);
/* результат выполнения
   MetaQuotes-Demo 1889998: плечо = 1:100
   MetaQuotes-Demo 1889998: размер собственных средств на счете = 22139.86 USD
   MetaQuotes-Demo 1889998: текущий результат по открытым позициям = +174.00 USD
   EURUSD: значение одного пункта  = 0.00001
   EURUSD: текущий спред в пунктах = 12
   DBL_MAX = 1.79769313486231570e+308
   EMPTY_VALUE = 1.79769313486231570e+308
   PrintFormat(EMPTY_VALUE) = 1.797693e+308
   Print(EMPTY_VALUE) = 1.797693134862316e+308
*/
  }
```

Voir aussi

StringFormat, DoubleToString, Les types matériels (double, float)

© 2000-2024, MetaQuotes Ltd.

ResetLastError

Met la valeur de la variable prédéterminée _LastError dans le zéro.

```
void  ResetLastError();
```

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Il est nécessaire de noter que la fonction GetLastError() ne remet pas au zéro la valeur _LastError. D'habitude l'appel de la fonction est produit avant l'appel de la fonction, après laquelle l'apparition de l'erreurest vérifiée.

© 2000-2024, MetaQuotes Ltd.

Enter topic text here.

ResourceCreate

Crée la ressource de l'image à la base de l'ensemble des données. Il y a 2 variantes de la fonction:

La création de la ressource à la base du fichier

```
bool  ResourceCreate(
   const string      resource_name,       // le nom de la ressource
   const string      path                 // le chemin relatif au fichier
   );
```

La création de la ressource à la base du tableau des pixels

```
bool  ResourceCreate(
   const string      resource_name,       // le nom de la ressource
   const uint&       data[],              // un ensemble de données dans un tableau
   uint              img_width,           // la largeur de l'image - la ressource crée
   uint              img_height,          // la hauteur de l'image - la ressource crée
   uint              data_xoffset,        // le déplacement de l'angle gauche supérieu
   uint              data_yoffset,        // le déplacement de l'angle gauche supérieu
   uint              data_width,          // la largeur totale de l'image basée sur un
   ENUM_COLOR_FORMAT color_format         //la largeur totale de l'image basée sur un
   );
```

Paramètres

resource_name

[in] Nom de la ressource.

data[][]

[in] Un tableau unidimensionnel ou bidimensionnel pour créer une image complète.

img_width

[in] La largeur du domaine rectangulaire de l'image en pixels pour la mise dans la ressource en forme de l'image. Plus de valeur ne peut pas être*data_width*.

img_height

[in] La hauteur du domaine rectangulaire de l'image en pixels pour la mise dans la ressource en forme de l'image.

data_xoffset

[in] Le déplacement en pixels du domaine rectangulaire de l'image à l'horizontale à droite.

data_yoffset

[in] Le déplacement en pixels du domaine rectangulaire de l'image selon la verticale en bas.

data_width

[in] Il est nécessaire seulement pour les tableaux unidimentionnels et signifie la largeur complète de l'image créée de l'ensemble de données. Si *data_width*=0, il est supposé égal à *img_width*. Pour les tableaux bidimensionnels ce paramètre est ignoré et est accepté comme égal à la deuxième dimension du tableau *data[]*.

color_format

[in] Le moyen du traitement de la couleur de l'énumération ENUM_COLOR_FORMAT.

La valeur rendue

© 2000-2024, MetaQuotes Ltd.

true – en cas du succès, autrement false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError(). Les erreurs possibles:

- 4015 – ERR_RESOURCE_NAME_DUPLICATED (la coïncidence des noms des ressourcesstatiqueet dynamique),
- 4016 – ERR_RESOURCE_NOT_FOUND (la ressource n'est pas trouvée),
- 4017 – ERR_RESOURCE_UNSUPPORTED_TYPE (le type de la ressource n'est pas soutenu),
- 4018 – ERR_RESOURCE_NAME_IS_TOO_LONG (le nom de la ressource est trop long).

Note

Si la deuxième variante de la fonction est appelée pour créer la même ressource avec de différents paramètres de la largeur, de la hauteur et du décalage, une nouvelle ressource n'est pas créée de nouveau, mais on met à jour une ressource existante tout simplement.

La première variante de la fonction permet de charger les images et les sons des fichiers, la deuxième variante est destinée seulement à la création dynamique des images.

Les images doivent être au format BMP avec une profondeur de couleur de 24 ou 32 bits, le son peut être seulement au format WAV. Le montant du fichier de la ressource ne peut pas être plus de 16 Mb.

ENUM_COLOR_FORMAT

Identificateur	La description
COLOR_FORMAT_XRGB_NOALPHA	Le composant du canal alpha est ignoré
COLOR_FORMAT_ARGB_RAW	Les composants de la couleur ne sont pas traités par le terminal (ils doivent être correctement spécifiés par l'utilisateur)
COLOR_FORMAT_ARGB_NORMALIZE	Les composants de la couleur sont traités par le terminal

Voir aussi

Les ressources, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2024, MetaQuotes Ltd.

ResourceFree

Supprime la ressource dynamiquement créée(libère la mémoire occupée par la ressource).

```
bool  ResourceFree(
   const string  resource_name      // le nom de la ressource
   );
```

Paramètres

resource_name

[in] Le nom dela ressource, doit commencer par "::".

La valeur rendue

true - en cas du succès, autrement false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

La fonction ResourceFree () permet au développeur du programme mql5 de maîtriser la consommation de la mémoire au travail actif avec les ressources. Les objets graphiques, attachés à la ressource supprimée de la mémoire, s'afficheront correctement aussi après son suppression. Mais les objets graphiques créés de nouveau (OBJ_BITMAP et OBJ_BITMAP_LABEL) ne pourront pas utiliser la ressource supprimée.

La fonction supprime seulement les ressources dynamiques créées par ce programme.

Voir aussi

Les ressources, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2024, MetaQuotes Ltd.

ResourceReadImage

Lit les données de la ressource graphique, créé par la fonction ResourceCreate() ou sauvegardée dans le fichier EX5 à la compilation.

```
bool  ResourceReadImage(
   const string      resource_name,       // le nom de la ressource graphique pour la
   uint&             data[],              // le tableau pour la réception des données
   uint&             width,               // pour la réception de la largeur de l'imag
   uint&             height,              // pour la réception de la hauteur de l'imag
   );
```

Paramètres

resource_name

[in] Le nom de la ressource graphique contenant l'image. Pour l'accès aux ressources personnelles on indique court "::resourcename". S'il est nécessaire de charger la ressource du fichier EX5 compilé, il est nécessaire le nom dans l'aspect complet avec l'indication de la voie relativement le dossier MQL5, le nom du fichier et le nom de la ressource- "path\\filename.ex5::resourcename".

data[][]

[in] Un tableau unidimentionnel ou à deux dimensions pour la réception des données de la ressource graphique.

img_width

[out] La largeur de l'image de la ressource graphique en pixels .

img_height

[out] La hateur de l'image de la ressource graphique en pixels .

La valeur rendue

true - en cas du succès, autrement false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

Si à la base du tableau*data[]* il est nécessairede créer après la ressource graphique, il faut utiliser le format de la couleur COLOR_FORMAT_ARGB_NORMALIZE ou COLOR_FORMAT_XRGB_NOALPHA.

Si le tableau*data[]* est bidimensionnel et sa deuxième dimension est moins que la taille X(width) de la ressource graphique, la fonction ResourceReadImage() rendra false et la lecture ne sera pas produite. Mais en cela, si la ressource existe, les tailles actuelles de l'image reviennent aux paramètres width et height. Cela permettra de faire encore une tentative de la réception des données de la ressource.

Voir aussi

Les ressources, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2024, MetaQuotes Ltd.

ResourceSave

Sauvegarde la ressource au fichier spécifié.

```
bool  ResourceSave(
   const string  resource_name      // le nom de la ressource
   const string  file_name          // le nom du fichier
   );
```

Paramètres

resource_name

[in] Le nom de la ressource doit commencer par "::".

file_name

[in] Le nom du fichier relativement MQL5\Files.

La valeur rendue

true – si en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

La fonction réenregistre toujours le fichier et en cas de nécessité crée tous les sous-répertoires intermédiaires dans le nom du fichier en cas de leur absence.

Voir aussi

Les ressources, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2024, MetaQuotes Ltd.

SetReturnError

Définit le code que le process du terminal retourne à la fin d'une opération.

```
void  SetReturnError(
   int ret_code      // code de fin du terminal client
   );
```

Parameters

ret_code

[in] Le code que le process du terminal client doit retourner à la fin de l'opération.

Valeur de Retour

Aucune valeur de retour.

Note

Définir le code de retour *ret_code* spécifié avec la fonction SetReturnError() est utile pour analyser les raisons de la fin de l'opération lors du lancement du terminal via la ligne de commande.

Contrairement à TerminalClose(), la fonction SetReturnError() ne termine pas l'opération du terminal. Elle définit au lieu de cela le code que le process du terminal retournera à la fin.

Si la fonction SetReturnError() est appelée plusieurs fois, et/ou depuis différents programmes MQL5, le terminal retourne le dernier code de retour défini.

Le code défini est retourné à la fin du process du terminal, sauf dans les cas suivants :

- une erreur critique est survenue pendant l'exécution ;
- la fonction TerminalClose(int ret_code) générant la commande de fin du terminal avec le code spécifié a été appelé.

Voir également

Exécution du programme, Erreurs d'Exécution, Codes de Désinitialisation, TerminalClose

© 2000-2024, MetaQuotes Ltd.

SetUserError

Établit la variable prédéterminée _LastError à la valeur égale à ERR_USER_ERROR_FIRST + user_error

```
void  SetUserError(
   ushort user_error,   // numéro de l'erreur
   );
```

Paramètres

user_error

[in] Le numero de l'erreur , établi par l'utilisateur.

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Après qu'une erreur a été mise à l'aide de la fonction SetUserError(user_error), la fonction GetLastError() rendra la valeur, égal à ERR_USER_ERROR_FIRST + user_error.

Exemple:

```
void OnStart()
  {
//--- définissons le numéro de l'erreur 65537=(ERR_USER_ERROR_FIRST +1)
   SetUserError(1);
//--- recevrons le code de la dernière erreur
   Print("GetLastError = ",GetLastError());
/*
   Résultat
   GetLastError = 65537
*/
  }
```

© 2000-2024, MetaQuotes Ltd.

Sleep

Retient l'exécution de l'expert courant ou le script sur l'intervalle défini.

```
void  Sleep(
   int  milliseconds      // intervalle
   );
```

Paramètres

milliseconds

[in] L'interligne du retard des millisecondes.

La valeur rendue

Il n'y a pas de valeur rendue.

Note

On ne peut pas appeler la fonction Sleep() des indicateurs d'utilisateur, puisque les indicateurs sont accomplis dans le thread d'interface et ne doivent pas le ralentir. La vérification de l'état du drapeau du stoppage de l'expert de chaque 0.1 secondes est inseré dans la fonction.

© 2000-2024, MetaQuotes Ltd.

TerminalClose

Envoie au terminal l'ordre sur l'achèvement du travail.

```
bool  TerminalClose(
   int ret_code      // code de retour du terminal de client
   );
```

Paramètres

ret_code

[in] Le code de retour rendu par le procès du terminal de client à l'achèvement du travail.

La valeur rendue

Rend true en cas du succès, autrement false.

Note

La fonction TerminalClose() ne produit pas l'arrêt immédiat du travail du terminal, elle envoie simplement au terminal la commande sur l'achèvement.

Dans le code du conseiller qui a appelé TerminalClose(), on doit faire toutes les préparations pour l'achèvement immédiat du travail (par exemple, tous les fichiers auparavant ouverts doivent être fermés). Après l'appel de cette fonction va l'opérateur return.

Le paramètre *ret_code* permet d'indiquer le code de retour nécessaire pour l'analyse des raisons de la cessation du travail de programme du terminal à son lancement de la ligne d'instruction.

Exemple:

```
//--- input parameters
input int  tiks_before=500; // quantité de ticks avant l'achèvement
input int  pips_to_go=15;   // distance aux pips
input int  seconds_st=50;   // combien de secondes donnons-nous à l'expert
//--- globals
datetime   launch_time;
int        tick_counter=0;
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(__FUNCTION__," reason code = ",reason);
   Comment("");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   static double first_bid=0.0;
```

© 2000-2024, MetaQuotes Ltd.

```
   MqlTick       tick;
   double        distance;
//---
   SymbolInfoTick(_Symbol,tick);
   tick_counter++;
   if(first_bid==0.0)
     {
      launch_time=tick.time;
      first_bid=tick.bid;
      Print("first_bid = ",first_bid);
      return;
     }
//--- marche du prix dans les points
   distance=(tick.bid-first_bid)/_Point;
//--- montrons le message pour surveiller au travail de conseiller
   string comm="Après le moment du lancement :\r\n\x25CF a passé des secondes: "+
               IntegerToString(tick.time-launch_time)+" ;"+
               "\r\n\x25CF ticks reçues: "+(string)tick_counter+" ;"+
               "\r\n\x25CF prix a passé dans les points: "+StringFormat("%G",distance)
   Comment(comm);
//--- section de la vérification des conditions de la clôture du terminal
   if(tick_counter>=tiks_before)
      TerminalClose(0);    // sortie selon le compteur des ticks
   if(distance>pips_to_go)
      TerminalClose(1);    // ont passé en haut aux pips_to_go pips
   if(distance<-pips_to_go)
      TerminalClose(-1);   // ont passé en bas aux pips_to_go pips
   if(tick.time-launch_time>seconds_st)
      TerminalClose(100);  // achèvement du travail selon le timeout
//---
  }
```

Voir aussi

L'exécution des programmes, Les erreurs de l'exécution, Les raisons de la déinitialisation

© 2000-2024, MetaQuotes Ltd.

TesterHideIndicators

Définit le mode d'affichage des indicateurs utilisés dans un EA. La fonction permet de gérer la visibilité des indicateurs utilisés pendant le test uniquement.

```
void  TesterHideIndicators(
   bool      hide     // flag
   );
```

Paramètres

hide

[in] Flag pour cacher les indicateurs lors d'un test. Indiquez true pour cacher les indicateurs créés, false sinon.

Valeur de Retour

Aucune.

Note

Par défaut, tous les indicateurs créés dans un EA testé sont affichés sur le graphique du test. En outre, ces indicateurs sont affichés sur la graphique qui est ouvert automatiquement à la fin du test. La fonction TesterHideIndicators() permet aux développeurs d'implémenter la possibilité de désactiver l'affichage des indicateurs utilisés.

Pour désactiver l'affichage d'un indicateur appliqué lors du test d'un EA, appelez TesterHideIndicators() avec true avant de créer le handle de l'EA - tous les indicateurs créés après cela sont marqués avec le flag. Ces indicateurs ne sont pas affichés pendant un test visuel et sur le graphique ouvert automatiquement à la fin du test.

Pour désactiver le mode masqué des nouveaux indicateurs, appelez TesterHideIndicators() avec false. Seuls les indicateurs générés directement à partir de l'EA testé peuvent être affichés sur le graphique du test. Cette règle ne s'applique qu'aux cas où il n'y a pas de modèle dans <data_folder>MQL5\Profiles\Templates.

Si le répertoire <data_folder>MQL5\Profiles\Templates contient un modèle <nom_EA>.tpl, seuls les indicateurs de ce modèle seront affichés pendant un test visuel et sur le graphique du test. Dans ce cas, aucun indicateur appliqué à l'EA testé n'est affiché. Ce comportement reste même si TesterHideIndicators() avec true est appelé dans le code de l'EA.

Si le répertoire <data_folder>MQL5\Profiles\Templates ne contient pas de modèle <nom_EA>.tpl ayant au lieu de cela tester.tpl, les indicateurs de tester.tpl et ceux de l'EA non désactivés par la fonction TesterHideIndicators() sont affichés pendant un test visuel et sur le graphique du test. S'il n'y a pas de modèle tester.tpl, les indicateurs du modèle default.tpl sont utilisés.

Si le strategy tester ne trouve aucun modèle convenable (<nom_EA>.tpl, tester.tpl ou default.tpl), l'affichage des indicateurs appliqués dans l'EA est intégralement géré par la fonction TesterHideIndicators().

Exemple :

```
bool CSampleExpert::InitIndicators(void)
  {
   TesterHideIndicators(true);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- crée l'indicateur MACD
   if(m_handle_macd==INVALID_HANDLE)
      if((m_handle_macd=iMACD(NULL,0,12,26,9,PRICE_CLOSE))==INVALID_HANDLE)
        {
         printf("Erreur lors de la création de l'indicateur MACD");
         return(false);
        }
   TesterHideIndicators(false);
//--- crée un indicateur EMA et l'ajoute à la collection
   if(m_handle_ema==INVALID_HANDLE)
      if((m_handle_ema=iMA(NULL,0,InpMATrendPeriod,0,MODE_EMA,PRICE_CLOSE))==INVALID_H
        {
         printf("Erreur lors de la création de l'indicateur EMA");
         return(false);
        }
//--- succès
   return(true);
  }
```

Voir aussi

IndicatorRelease

© 2000-2024, MetaQuotes Ltd.

Enter topic text here.

TesterStatistics

Rend la valeur du paramètre indiqué statistique, calculée d'après les résultats du test

```
double  TesterStatistics(
   ENUM_STATISTICS statistic_id      // l'identificateur
   );
```

Paramètres

statistic_id

[in] L'identificateur du paramètre statistique de l'énumérationENUM_STATISTICS.

La valeur rendue

La valeur du paramètre statistique des résultats du test.

Note

La fonction peut être appelée au sein d' OnTester () ou OnDeinit () dans le testeur. Dans d'autres cas, le résultat est indéfini.

© 2000-2024, MetaQuotes Ltd.

TesterStop

Envoie la commande de fin du programme lors d'un test.

```
void  TesterStop();
```

Valeur de Retour

Aucune valeur de retour.

Note

La fonction TesterStop() est conçue pour la fin prématurée d'un EA sur un agent de test - par exemple, lorsqu'un nombre spécifié de trades perdants ou un niveau de drawdown sont atteints.

Un appel à TesterStop() est considéré comme étant une fin normale d'un test, la fonction OnTester() est donc appelée et les statistiques complètes de trading ainsi que la valeur du critère d'optimisation sont renvoyées au strategy tester.

Un appel à ExpertRemove() dans le strategy tester signifie également la fin normale du test et permet d'obtenir les statistiques de trading, mais l'EA est enlevé de la mémoire de l'agent. Dans ce cas, effectuer une passe de l'ensemble suivant de paramètres nécéssite plus de temps afin de recharger le programme. TesterStop() est donc l'option préférée pour une fin prématurée d'un test.

Voir également

Exécution du Programme, Tester des Stratégies de Trading, ExpertRemove, SetReturnError

© 2000-2024, MetaQuotes Ltd.

TesterDeposit

Fonction spéciale qui émule le dépôt de fonds pendant un test. Elle peut être utilisée dans certains système de money management.

```
bool  TesterDeposit(
   double money      // somme déposée
   );
```

Paramètres

money

[in] Somme d'argent à déposer sur le compte dans la devise du dépôt.

Valeur de Retour

Retourne true en cas de succès, sinon retourne false.

Voir aussi

TesterWithdrawal

© 2000-2024, MetaQuotes Ltd.

TesterWithdrawal

La fonction spéciale pour l'émulation des opérations du main levée des moyens pendant le test. Peut être utilisé dans certains systèmes de gestion du capital.

```
bool  TesterWithdrawal(
   double money      // montant de la somme retirée
   );
```

Les paramètres

money

[in] Le montant des moyens de financement, dont il est nécessaire de retirer du compte (en devise du dépôt).

La valeur rendue

Rend true en cas du succès, autrement rend false.

Voir aussi

TesterDeposit

© 2000-2024, MetaQuotes Ltd.

TranslateKey

Retourne le caractère Unicode d'une touche de touche virtuel suivant la langue actuelle de l'utilisateur et le statut des touches de contrôle.

```
short  TranslateKey(
   int  key_code      // code de touche pour lequel on veut récupérer le caractère Uni
   );
```

Paramètres

key_code

[in] Code de la touche.

Valeur de Retour

Le caractère Unicode dans le cas d'une conversion réussie. La fonction retourne -1 en cas d'erreur.

Note

La fonction utilise ToUnicodeEx pour convertir en caractères Unicode les touches sur lesquelles l'utilisateur a appuyé. Une erreur peut survenir dans le cas où ToUnicodeEx n'est pas déclenchée - par exemple, lors de la réception du caractère de la touche SHIFT.

Exemple:

```
void OnChartEvent(const int id,const long& lparam,const double& dparam,const string& s
  {
   if(id==CHARTEVENT_KEYDOWN)
     {
      short sym=TranslateKey((int)lparam);
      //--- si le caractère saisi est correctement converti en Unicode
      if(sym>0)
         Print(sym,"'",ShortToString(sym),"'");
      else
         Print("Erreur dans TranslateKey pour la touche =",lparam);
     }
  }
```

Voir aussi

Evènements du terminal client, OnChartEvent

© 2000-2024, MetaQuotes Ltd.

ZeroMemory

Remet au zéro la variable transmise selon la référence.

```
void  ZeroMemory(
   void & variable      // variable de reconstruction
   );
```

Paramètres

variable

[in] [out] La variable transmise selon le lien laquelle il faut remettre au zéro (initialisez par les valeurs zéro).

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Si le paramètre de la fonction est la ligne, cet appel sera équivalent à l'indication pour elle de la valeur NULL.
Pour les types simples et leurs tableaux, ainsi que les structures/classes comprenant tels types, c'est un simple remise au zéro.
Pour les objets contenant les lignes et les tableaux dynamiques, on produit l'appel ZeroMemory() pour chaque membre.
Pour chaque tableaux non protégés par le modificateur const, la remise au zéro de tous les éléments est faite.
Pour les tableaux des objets complexes on appelle ZeroMemory() pour chaque élément.

La fonction ZeroMemory() n'est pas employée pour les classes avec les membres protégés ou l'héritage.

© 2000-2024, MetaQuotes Ltd.

Le groupe des fonctions pour le travail avec les tableaux

Pas plus que les tableaux quadridimensionnels sont admissibles. L'indexation de chaque mesure est produite de 0 jusqu'à *la grandeur_de la dimension-1*. Dans le cas particulier de tableaux unidimentionnel des 50 éléments l'appel au premier élément aura l'air comme array[0], au dernier élément - array[49].

Fonction	Action
ArrayBsearch	Rend l'index du premier élément trouvé dans dans la première dimension du tableau
ArrayCopy	Copie un tableau à l'autre
ArrayCompare	Rend le résultat de la comparaison de deux tableux des types simples ou des structures d'utilisateur qui n'ont pas des objets complexes
ArrayFree	Libère la mémoire du tampon de n'importe quel tableau dynamique et met la dimension de la valeur zéro au 0.
ArrayGetAsSeries	Contrôle la direction de l'indexation du tableau
ArrayInitialize	Met tous les éléments du tableau numérique à une valeur
ArrayFill	Remplit le tableau numérique par la valeur indiquée
ArrayIsSeries	Vérifies - si le tableau est la série temporelle
ArrayIsDynamic	Vérifies - si le tableau est dynamique
ArrayMaximum	La recherche de l'élément avec la valeur maximal
ArrayMinimum	La recherche de l'élément avec la valeur minimal
ArrayPrint	Ecrit un tableau de types simples ou une structure simple dans le journal
ArrayRange	Rend le nombre d'éléments dans la dimension indiquée du tableau
ArrayResize	Met la nouvelle grandeur dans la première dimension du tableau
ArrayInsert	Insère le nombre spécifié d'éléments d'un tableau source dans un tableau destination à partir d'un indice spécifié
ArrayRemove	Supprime le nombre spécifié d'éléments du tableau à partir de l'indice spécifié
ArrayReverse	Inverse le nombre spécifié d'éléments dans le tableau en commencant à l'indice spécifié
ArraySetAsSeries	Établit la direction de l'indexation dans le tableau
ArraySize	Rend le nombre d'éléments dans le tableau
ArraySort	Le triage des tableaux numériques par la première dimension
ArraySwap	Intervertit les contenus de deux tableaux dynamiques du même type

© 2000-2024, MetaQuotes Ltd.

ArrayBsearch

Recherche une valeur spécifiée dans un tableau numérique multi-dimmensionnel trié par ordre croissant. La recherche est effectuée parmi les éléments de la première dimension.

Pour chercher dans un tableau d'éléments de type double

```
int  ArrayBsearch(
   const double&    array[],   // tableau où effectuer la recherche
   double           value      // valeur à rechercher
   );
```

Pour chercher dans un tableau d'éléments de type float

```
int  ArrayBsearch(
   const float&    array[],   // tableau où effectuer la recherche
   float           value      // valeur à rechercher
   );
```

Pour chercher dans un tableau d'éléments de type long

```
int  ArrayBsearch(
   const long&    array[],   // tableau où effectuer la recherche
   long           value      // valeur à rechercher
   );
```

Pour chercher dans un tableau d'éléments de type int

```
int  ArrayBsearch(
   const int&    array[],   // tableau où effectuer la recherche
   int           value      // valeur à rechercher
   );
```

Pour chercher dans un tableau d'éléments de type short

```
int  ArrayBsearch(
   const short&    array[],   // tableau où effectuer la recherche
   short           value      // valeur à rechercher
   );
```

Pour chercher dans un tableau d'éléments de type caractères

```
int  ArrayBsearch(
   const char&    array[],   // tableau où effectuer la recherche
   char           value      // valeur à rechercher
   );
```

Paramètres

array[]

[in] Tableau numérique ou effectuer la recherche.

value

© 2000-2024, MetaQuotes Ltd.

[in] Valeur à rechercher.

Valeur de Retour

La fonction retourne l'indice de l'élément trouvé. Si la valeur recherchée n'est pas trouvée, la fonction retourne l'indice de l'élément dont la valeur est la plus proche.

Note

La recherche binaire ne s'effectue que sur des tableaux triés. Pour trier des tableaux numériques, utiliser la fonction ArraySort().

Exemple :

```
#property description "Le script, basé sur l'indicateur RSI, affiche"
#property description "combien de fois le marché était dans les zones"
#property description "de surachat/survente dans l'intervalle de temps spécifié."
//--- affiche la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- paramètres d'entrée
input int                 InpMAPeriod=14;                    // Période de la moyenne r
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;       // Type du prix
input double              InpOversoldValue=30.0;            // Niveau de survente
input double              InpOverboughtValue=70.0;          // Niveau de surachat
input datetime            InpDateStart=D'2012.01.01 00:00';  // Date de début d'analyse
input datetime            InpDateFinish=D'2013.01.01 00:00'; // Date de fin d'analyse
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   double rsi_buff[]; // tableau des valeurs de l'indicateur
   int    size=0;     // taille du tableau
//--- récupère le handle de l'indicateur RSI
   ResetLastError();
   int rsi_handle=iRSI(Symbol(),Period(),InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- échec de récupération du handle de l'indicateur
      PrintFormat("Erreur de récupération du handle de l'indicateur. Code d'erreur = %
      return;
     }
//--- boucle tant que toutes les valeurs de l'indicateur n'ont pas été calculées
   while(BarsCalculated(rsi_handle)==-1)
     {
      //--- sort si l'indicateur a été arrêté
      if(IsStopped())
         return;
      //--- pause pour permettre à l'indicateur de calculer toutes ses valeurs
      Sleep(10);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- copie les valeurs de l'indicateur pour un certain intervalle de temps
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,InpDateFinish,rsi_buff)==-1)
     {
      PrintFormat("Echec de la copie des valeurs de l'indicateur. Code d'erreur = %d",
      return;
     }
//--- récupère la taille du tableau
   size=ArraySize(rsi_buff);
//--- trie le tableau
   ArraySort(rsi_buff);
//--- recherche du temps passé (en pourcentage) par le marché en zone de survente
   double ovs=(double)ArrayBsearch(rsi_buff,InpOversoldValue)*100/(double)size;
//--- recherche du temps passé (en pourcentage) par le marché en zone de surachat
   double ovb=(double)(size-ArrayBsearch(rsi_buff,InpOverboughtValue))*100/(double)siz
//--- crée les chaînes de caractères pour l'affichage des données
   string str="De "+TimeToString(InpDateStart,TIME_DATE)+" à "
              +TimeToString(InpDateFinish,TIME_DATE)+" le marché était :";
   string str_ovb="en zone de surachat "+DoubleToString(ovb,2)+"% du temps";
   string str_ovs="en zone de survente "+DoubleToString(ovs,2)+"% du temps";
//--- affiche les données sur le graphique
   CreateLabel("top",5,60,str,clrDodgerBlue);
   CreateLabel("suracheté",5,35,str_ovb,clrDodgerBlue);
   CreateLabel("survendu",5,10,str_ovs,clrDodgerBlue);
//--- redéssine le graphique
   ChartRedraw(0);
//--- pause
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Affiche le commentaire dans le coin inférieur gauche du graphique|
//+------------------------------------------------------------------+
void CreateLabel(const string name,const int x,const int y,
                 const string str,const color clr)
  {
//--- crée l'étiquette
   ObjectCreate(0,name,OBJ_LABEL,0,0,0);
//--- lie le lable au coin inférieur gauche
   ObjectSetInteger(0,name,OBJPROP_CORNER,CORNER_LEFT_LOWER);
//--- change la position du point d'ancrage
   ObjectSetInteger(0,name,OBJPROP_ANCHOR,ANCHOR_LEFT_LOWER);
//--- distance depuis le point d'ancrage dans la direction X
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
//--- distance depuis le point d'ancrage dans la direction Y
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
//--- texte de l'étiquette
   ObjectSetString(0,name,OBJPROP_TEXT,str);
//--- couleur du texte
   ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- taille du texte
   ObjectSetInteger(0,name,OBJPROP_FONTSIZE,12);
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayCopy

Copie un tableau dans un autre.

```
int  ArrayCopy(
   void&        dst_array[],         // tableau de destination
   const void&  src_array[],         // tableau source
   int          dst_start=0,         // indice de début d'écriture dans le tableau des
   int          src_start=0,         // premier indice du tableau source
   int          count=WHOLE_ARRAY    // nombre d'éléments
   );
```

Paramètres

dst_array[]

[out] Tableau de destination

src_array[]

[in] Tableau source

dst_start=0

[in] Indice de début dans le tableau de destination. Pa défaut, l'indice de début est 0.

src_start=0

[in] Indice de début pour le tableau source. Pa défaut, l'indice de début est 0.

count=WHOLE_ARRAY

[in] Nombre d'éléments devant être copiés. Par défaut, le tableau est copié en entier (count=WHOLE_ARRAY).

Valeur de Retour

Retourne le nombre d'éléments copiés.

Note

Si count<0 ou count>src_size-src_start, toute la partie restante du tableau est copiée. Les tableaux sont copiés de la gauche vers la droite. Pour les tableaux de type séries, la position de départ est correctement ajustée pour copier de la gauche vers la droite.

Si les tableaux sont de types différentes, la fonction essaye pendant la copie de transformer chaque élément du tableau source dans le type du tableau de destination. Un tableau de chaînes de caractères ne peut être copié que dans un autre tableau de chaînes de caractères. Un tableau de classes et de structures contenant des objets nécessitant une initialisation n'est pas copié. Un tableau de structures ne peut être copié que dans un tableau du même type.

Pour les tableaux dynamiques avec une indexation comme les timeseries, la taille du tableau de destination est automatiquement augmentée jusqu'au nombre de données copiées (si ce dernier est supérieur à la taille du tableau). La taille du tableau de destination n'est pas diminuée automatiquement.

Exemple :

```
#property description "L'indicateur met en lumière les bougies qui sont des "
```

© 2000-2024, MetaQuotes Ltd.

```
#property description "plus hauts et des plus bas locaux. La longueur de l'intervalle
#property description "les valeurs extrêmes est disponible dans les paramètres d'entré
//--- paramètres de l'indicateur
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- éléments dessinés
#property indicator_label1  "Extremums"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrLightSteelBlue,clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- constantes prédéfinies
#define INDICATOR_EMPTY_VALUE 0.0
//--- paramètres d'entrée
input int InpNum=4; // Longueur d'une moitié d'intervalle
//--- buffers de l'indicateur
double ExtOpen[];
double ExtHigh[];
double ExtLow[];
double ExtClose[];
double ExtColor[];
//--- variables globales
int    ExtStart=0; // indice de la première bougier qui n'est pas un extrême
int    ExtCount=0; // nombre de non-extrêmes dans l'intervalle
//+------------------------------------------------------------------+
//| Remplissage des bougies non-extrêmes                             |
//+------------------------------------------------------------------+
void FillCandles(const double &open[],const double &high[],
                 const double &low[],const double &close[])
  {
//--- remplit les bougies
   ArrayCopy(ExtOpen,open,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtHigh,high,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtLow,low,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtClose,close,ExtStart,ExtStart,ExtCount);
  }
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,ExtOpen);
   SetIndexBuffer(1,ExtHigh);
   SetIndexBuffer(2,ExtLow);
   SetIndexBuffer(3,ExtClose);
   SetIndexBuffer(4,ExtColor,INDICATOR_COLOR_INDEX);
//--- spécifie la valeur, qui n'est pas affichée
```

© 2000-2024, MetaQuotes Ltd.

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- spécifie les noms des buffers de l'indicateur pour les afficher dans la fenêtre
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- définit l'indexation des timeseries
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- variable du départ du calcul de la barre
   int start=prev_calculated;
//--- le calcul n'est pas effectué pour les InpNum*2 premières barres
   if(start==0)
     {
      start+=InpNum*2;
      ExtStart=0;
      ExtCount=0;
     }
//--- si la barre vient juste de se former, vérifie l'extrême potentiel suivant
   if(rates_total-start==1)
      start--;
//--- indice de barre à vérifier pour l'extrême
   int ext;
//--- boucle de calcul des valeurs de l'indicateur
   for(int i=start;i<rates_total-1;i++)
     {
      //--- initialement sur une barre i sans dessin
      ExtOpen[i]=0;
      ExtHigh[i]=0;
      ExtLow[i]=0;
      ExtClose[i]=0;
      //--- indice d'extrême pour vérification
      ext=i-InpNum;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- vérifie le maximum local
      if(IsMax(high,ext))
        {
         //--- met en avant la bougie d'extrême
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=1;
         //--- met en avant les autres bougies jusqu'à l'extrême avec une couleur neut
         FillCandles(open,high,low,close);
         //--- change les couleurs de la variable
         ExtStart=ext+1;
         ExtCount=0;
         //--- passe à l'itération suivante
         continue;
        }
      //--- vérifie le minimum local
      if(IsMin(low,ext))
        {
         //--- met en avant la bougie d'extrême
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=2;
         //--- met en avant les autres bougies jusqu'à l'extrême avec une couleur neut
         FillCandles(open,high,low,close);
         //--- change les valeurs de la variable
         ExtStart=ext+1;
         ExtCount=0;
         //--- passe à l'itération suivante
         continue;
        }
      //--- augmente le nombre de non-extrêmes dans l'intervalle
      ExtCount++;
     }
//--- retourne la valeur de prev_calculated pour le prochain appel
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Vérifie si l'élément actuel du tableau est un plus haut local    |
//+------------------------------------------------------------------+
bool IsMax(const double &price[],const int ind)
  {
//--- variable de début de l'intervalle
   int i=ind-InpNum;
//--- période de fin de l'intervalle
   int finish=ind+InpNum+1;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- vérifie la première moitié de l'intervalle
   for(;i<ind;i++)
     {
      if(price[ind]<=price[i])
         return(false);
     }
//--- vérifie la deuxième moitié de l'intervalle
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]<=price[i])
         return(false);
     }
//--- c'est un extrême
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie si l'élément actuel du tableau est un plus bas local     |
//+------------------------------------------------------------------+
bool IsMin(const double &price[],const int ind)
  {
//--- variable de début de l'intervalle
   int i=ind-InpNum;
//--- variable de fin de l'intervalle
   int finish=ind+InpNum+1;
//--- vérifie la première moitié de l'intervalle
   for(;i<ind;i++)
     {
      if(price[ind]>=price[i])
         return(false);
     }
//--- vérifie la deuxième moitié de l'intervalle
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]>=price[i])
         return(false);
     }
//--- c'est un extrême
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayCompare

La fonction retourne le résultat de la comparaison de deux tableaux du même type. Elle peut être utilisée pour comparer des tableaux d'éléments de types simples ou des structures personnalisées sans objets complexes, c'est à dire des structures ne contenant pas d'objets de type string, de tableaux dynamiques, de classes ou d'autres structures contenant des objets complexes.

```
int  ArrayCompare(
   const void&  array1[],            // premier tableau
   const void&  array2[],            // deuxième tableau
   int          start1=0,            // décalage initial du premier tableau
   int          start2=0,            // décalage initial du deuxième tableau
   int          count=WHOLE_ARRAY    // nombre d'éléments à comparer
   );
```

Paramètres

array1[]

[in] Premier tableau.

array2[]

[in] Deuxième tableau.

start1=0

[in] L'indice initial de l'élément dans le premier tableau, à partir duquel la comparaison commence. L'indice de début par défaut est 0.

start2=0

[in] L'indice initial de l'élément dans le deuxième tableau, à partir duquel la comparaison commence. L'indice de début par défaut est 0.

count=WHOLE_ARRAY

[in] Nombre d'éléments à comparer. Tous les éléments des deux tableaux participent par défaut à la comparaison (count=WHOLE_ARRAY).

Valeur de Retour

- -1, si array1[] est inférieur à array2[]
- 0, si array1[] est égal à array2[]
- 1, si array1[] est supérieur à array2[]
- -2, si une erreur est survenue en raison d'une incompatibilité des types des tableaux comparés ou si start1, start2 ou le nombre de valeurs comptées dépassent les limites du tableau.

Note

La fonction ne retournera pas 0 (les tableaux ne seront pas considérés comme égaux) si les tableaux diffèrent en taille et que count=WHOLE_ARRAY pour le cas où un tableau est un sous-ensemble de l'autre. Dans ce cas, le résultat de la comparaison des tailles de ces tableaux retournera : -1, si la taille de array1[] est inférieure à celle de array2[] , sinon 1.

© 2000-2024, MetaQuotes Ltd.

ArrayFree

La fonction libère le buffer d'un tableau dynamique et définit la taille de la dimension 0 (la première dimension) à 0.

```
void  ArrayFree(
   void&  array[]      // tableau
   );
```

Paramètres

array[]

[in] Tableau dynamique.

Valeur de Retour

Aucune valeur de retour.

Note

Le besoin d'utiliser la fonction ArrayFree() n'apparaît pas très souvent si l'on considère que toute la mémoire utilisée est libérée en une fois et que l'utilisation d'un tableau comprend l'accès aux buffers de l'indicateur. Les tailles des buffers sont automatiquement gérées par le système d'exécution du terminal.

S'il est nécessaire de gérer manuellement la mémoire dans un environnement dynamique complexe de l'application, la fonction ArrayFree() permet aux utilisateur de libérer la mémoire occupée par le tableau qui n'est plus nécessaire de façon explicite et immédiatement.

Exemple :

```
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
#include <Controls\Label.mqh>
#include <Controls\ComboBox.mqh>
//--- constantes prédéfinies
#define X_START 0
#define Y_START 0
#define X_SIZE 280
#define Y_SIZE 300
//+------------------------------------------------------------------+
//| Classe de dialogue pour travailler avec la mémoire               |
//+------------------------------------------------------------------+
class CMemoryControl : public CAppDialog
  {
private:
   //--- taille du tableau
   int               m_arr_size;
   //--- tableaux
   char              m_arr_char[];
   int               m_arr_int[];
   float             m_arr_float[];
```

© 2000-2024, MetaQuotes Ltd.

```
   double            m_arr_double[];
   long              m_arr_long[];
   //--- étiquettes
   CLabel            m_lbl_memory_physical;
   CLabel            m_lbl_memory_total;
   CLabel            m_lbl_memory_available;
   CLabel            m_lbl_memory_used;
   CLabel            m_lbl_array_size;
   CLabel            m_lbl_array_type;
   CLabel            m_lbl_error;
   CLabel            m_lbl_change_type;
   CLabel            m_lbl_add_size;
   //--- boutons
   CButton           m_button_add;
   CButton           m_button_free;
   //--- listes
   CComboBox         m_combo_box_step;
   CComboBox         m_combo_box_type;
   //--- valeur actuelle du type du tableau pour la liste
   int               m_combo_box_type_value;

public:
                     CMemoryControl(void);
                    ~CMemoryControl(void);
   //--- méthode de création de l'objet de classe
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènements du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crée les étiquettes
   bool              CreateLabel(CLabel &lbl,const string name,const int x,const int y
   //--- crée les éléments de contrôle
   bool              CreateButton(CButton &button,const string name,const int x,const
   bool              CreateComboBoxStep(void);
   bool              CreateComboBoxType(void);
   //--- gestionnaires d'évènements
   void              OnClickButtonAdd(void);
   void              OnClickButtonFree(void);
   void              OnChangeComboBoxType(void);
   //--- méthodes d'utilisation du tableau actuel
   void              CurrentArrayFree(void);
   bool              CurrentArrayAdd(void);
  };
//+------------------------------------------------------------------+
//| Libère la mémoire du tableau actuel                              |
//+------------------------------------------------------------------+
void CMemoryControl::CurrentArrayFree(void)
  {
```

```
//--- réinitialise la taille du tableau
   m_arr_size=0;
//--- libère le tableau
   if(m_combo_box_type_value==0)
      ArrayFree(m_arr_char);
   if(m_combo_box_type_value==1)
      ArrayFree(m_arr_int);
   if(m_combo_box_type_value==2)
      ArrayFree(m_arr_float);
   if(m_combo_box_type_value==3)
      ArrayFree(m_arr_double);
   if(m_combo_box_type_value==4)
      ArrayFree(m_arr_long);
  }
//+------------------------------------------------------------------+
//| Tentative d'ajout de mémoire au tableau actuel                   |
//+------------------------------------------------------------------+
bool CMemoryControl::CurrentArrayAdd(void)
  {
//--- sort si la taille de la mémoire utilisée est supérieure à la taille de la mémoi
   if(TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL)/TerminalInfoInteger(TERMINAL_MEMOI
      return(false);
//--- tentative d'allocation de la mémoire selon le type courant
   if(m_combo_box_type_value==0 && ArrayResize(m_arr_char,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==1 && ArrayResize(m_arr_int,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==2 && ArrayResize(m_arr_float,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==3 && ArrayResize(m_arr_double,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==4 && ArrayResize(m_arr_long,m_arr_size)==-1)
      return(false);
//--- mémoire allouée
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestion des évènements                                           |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CMemoryControl)
ON_EVENT(ON_CLICK,m_button_add,OnClickButtonAdd)
ON_EVENT(ON_CLICK,m_button_free,OnClickButtonFree)
ON_EVENT(ON_CHANGE,m_combo_box_type,OnChangeComboBoxType)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CMemoryControl::CMemoryControl(void)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CMemoryControl::~CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------+
//| Méthode de création d'un objet de la classe                      |
//+------------------------------------------------------------------+
bool CMemoryControl::Create(const long chart,const string name,const int subwin,
                            const int x1,const int y1,const int x2,const int y2)
  {
//--- crée un objet de la classe de base
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- prépare les chaînes de caractères des étiquettes
   string str_physical="Mémoire physique = "+(string)TerminalInfoInteger(TERMINAL_MEMO
   string str_total="Mémoire totale = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_TO
   string str_available="Mémoire disponible = "+(string)TerminalInfoInteger(TERMINAL_M
   string str_used="Mémoire utilisée = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_U
//--- crée les étiquettes
   if(!CreateLabel(m_lbl_memory_physical,"physical_label",X_START+10,Y_START+5,str_phy
      return(false);
   if(!CreateLabel(m_lbl_memory_total,"total_label",X_START+10,Y_START+30,str_total,12
      return(false);
   if(!CreateLabel(m_lbl_memory_available,"available_label",X_START+10,Y_START+55,str_
      return(false);
   if(!CreateLabel(m_lbl_memory_used,"used_label",X_START+10,Y_START+80,str_used,12,cl
      return(false);
   if(!CreateLabel(m_lbl_array_type,"type_label",X_START+10,Y_START+105,"Type de table
      return(false);
   if(!CreateLabel(m_lbl_array_size,"size_label",X_START+10,Y_START+130,"Array size =
      return(false);
   if(!CreateLabel(m_lbl_error,"error_label",X_START+10,Y_START+155,"",12,clrRed))
      return(false);
   if(!CreateLabel(m_lbl_change_type,"change_type_label",X_START+10,Y_START+185,"Chang
      return(false);
   if(!CreateLabel(m_lbl_add_size,"add_size_label",X_START+10,Y_START+210,"Add to arra
      return(false);
//--- crée les éléments de contrôle
   if(!CreateButton(m_button_add,"add_button",X_START+15,Y_START+245,"Add",12,clrBlue)
      return(false);
   if(!CreateButton(m_button_free,"free_button",X_START+75,Y_START+245,"Free",12,clrBl
      return(false);
   if(!CreateComboBoxType())
      return(false);
   if(!CreateComboBoxStep())
      return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- initialise la variable
   m_arr_size=0;
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée le bouton                                                   |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateButton(CButton &button,const string name,const int x,
                                  const int y,const string str,const int font_size,
                                  const int clr)
  {
//--- crée le bouton
   if(!button.Create(m_chart_id,name,m_subwin,x,y,x+50,y+20))
      return(false);
//--- texte
   if(!button.Text(str))
      return(false);
//--- taille de la police de caractères
   if(!button.FontSize(font_size))
      return(false);
//--- couleur de l'étiquette
   if(!button.Color(clr))
      return(false);
//--- ajoute le bouton aux éléments de contrôle
   if(!Add(button))
      return(false);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée une liste pour la taille du tableau                         |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxStep(void)
  {
//--- crée la liste
   if(!m_combo_box_step.Create(m_chart_id,"step_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- ajoute des éléments à la liste
   if(!m_combo_box_step.ItemAdd("100 000",100000))
      return(false);
   if(!m_combo_box_step.ItemAdd("1 000 000",1000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("10 000 000",10000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("100 000 000",100000000))
      return(false);
//--- définit l'élément courant de la liste
   if(!m_combo_box_step.SelectByValue(1000000))
```

© 2000-2024, MetaQuotes Ltd.

```
      return(false);
//--- ajoute la liste aux éléments de contrôle
   if(!Add(m_combo_box_step))
      return(false);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée une liste pour le type du tableau                           |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxType(void)
  {
//--- crée la liste
   if(!m_combo_box_type.Create(m_chart_id,"type_combobox",m_subwin,X_START+100,Y_STARI
      return(false);
//--- ajoute des éléments à la liste
   if(!m_combo_box_type.ItemAdd("char",0))
      return(false);
   if(!m_combo_box_type.ItemAdd("int",1))
      return(false);
   if(!m_combo_box_type.ItemAdd("float",2))
      return(false);
   if(!m_combo_box_type.ItemAdd("double",3))
      return(false);
   if(!m_combo_box_type.ItemAdd("long",4))
      return(false);
//--- définit l'élément courant de la liste
   if(!m_combo_box_type.SelectByValue(3))
      return(false);
//--- stocke l'élément courant de la liste
   m_combo_box_type_value=3;
//--- ajoute la liste aux éléments de contrôle
   if(!Add(m_combo_box_type))
      return(false);
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée une étiquette                                               |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateLabel(CLabel &lbl,const string name,const int x,
                                 const int y,const string str,const int font_size,
                                 const int clr)
  {
//--- crée une étiquette
   if(!lbl.Create(m_chart_id,name,m_subwin,x,y,0,0))
      return(false);
//--- texte
   if(!lbl.Text(str))
```

© 2000-2024, MetaQuotes Ltd.

```
      return(false);
//--- taille de la police de caractères
   if(!lbl.FontSize(font_size))
      return(false);
//--- couleur
   if(!lbl.Color(clr))
      return(false);
//--- ajoute l'étiquette aux éléments de contrôle
   if(!Add(lbl))
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement de clic sur le bouton "Ajouter"         |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonAdd(void)
  {
//--- augmente la taille du tableau
   m_arr_size+=(int)m_combo_box_step.Value();
//--- tentative d'allocation mémoire pour le tableau actuel
   if(CurrentArrayAdd())
     {
      //--- mémoire allouée, affiche le statut courant à l'écran
      m_lbl_memory_available.Text("Mémoire disponible = "+(string)TerminalInfoInteger
      m_lbl_memory_used.Text("Mémoire utilisée = "+(string)TerminalInfoInteger(TERMINA
      m_lbl_array_size.Text("Taille du tableau = "+IntegerToString(m_arr_size));
      m_lbl_error.Text("");
     }
   else
     {
      //--- échec d'allocation de la mémoire, affiche le message d'erreur
      m_lbl_error.Text("Le tableau est trop grand, erreur !");
      //--- retourne la taille précédente du tableau
      m_arr_size-=(int)m_combo_box_step.Value();
     }
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement de clic sur le bouton "Libérer"         |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonFree(void)
  {
//--- libère la mémoire du tableau actuel
   CurrentArrayFree();
//--- affiche le statut actuel à l'écran
   m_lbl_memory_available.Text("Mémoire disponible = "+(string)TerminalInfoInteger(TER
   m_lbl_memory_used.Text("Mémoire utilisée = "+(string)TerminalInfoInteger(TERMINAL_M
   m_lbl_array_size.Text("Taille du tableau = 0");
   m_lbl_error.Text("");
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement de changement de la liste               |
//+------------------------------------------------------------------+
void CMemoryControl::OnChangeComboBoxType(void)
  {
//--- vérifie si le type du tableau a changé
   if(m_combo_box_type.Value()!=m_combo_box_type_value)
     {
      //--- libère la mémoire du tableau actuel
      OnClickButtonFree();
      //--- utilise un autre type de tableau
      m_combo_box_type_value=(int)m_combo_box_type.Value();
      //--- affiche le nouveau type de tableau à l'écran
      if(m_combo_box_type_value==0)
         m_lbl_array_type.Text("Type de tableau = char");
      if(m_combo_box_type_value==1)
         m_lbl_array_type.Text("Type de tableau = int");
      if(m_combo_box_type_value==2)
         m_lbl_array_type.Text("Type de tableau = float");
      if(m_combo_box_type_value==3)
         m_lbl_array_type.Text("Type de tableau = double");
      if(m_combo_box_type_value==4)
         m_lbl_array_type.Text("Type de tableau = long");
     }
  }
//--- Objet de type CMemoryControl
CMemoryControl ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée la boîte de dialogue
   if(!ExtDialog.Create(0,"MemoryControl",0,X_START,Y_START,X_SIZE,Y_SIZE))
      return(INIT_FAILED);
//--- lancement
   ExtDialog.Run();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de dé-initialisation de l'expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ExtDialog.Destroy(reason);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Fonction d'évènement graphique de l'expert                       |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayGetAsSeries

Vérifie la direction de l'indexation d'un tableau.

```
bool  ArrayGetAsSeries(
   const void&  array[]    // tableau à vérifier
   );
```

Paramètres

array

[in] Tableau à vérifier.

Valeur de Retour

Retourne true si le tableau spécifié a le flag AS_SERIES, c'est à dire que l'accès au tableau est effectué de l'arrière vers l'avant comme dans les timeseries. Les timeseries diffèrent des tableaux classiques par leur indexation : l'indexation des éléments des timeseries est effectuée de la fin vers le début (des données les plus récentes aux plus anciennes).

Note

Pour vérifier si un tableau est de type timeseries, utiliser la fonction ArrayIsSeries(). Les tableaux de prix passés en paramètres à la fonction OnCalculate() n'ont pas obligatoirement le même sens d'indexation que les timeseries. Le sens nécessaire d'indexation peut être défini avec la fonction ArraySetAsSeries().

Exemple :

```
#property description "L'indicateur calcule les valeurs absolues des différences entre
#property description "les prix Open et Close ou High et Low, les affichant dans une s
#property description "sous forme d'histrogramme."
//--- paramètres de l'indicateur
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- éléments dessinés
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- paramètres d'entrée
input bool InpAsSeries=true; // Sens d'indexation dans le buffer de l'indicateur
input bool InpPrices=true;   // Prix à utilise pour les calculs (true - Open,Close ; f
//--- buffer de l'indicateur
double ExtBuffer[];
//+------------------------------------------------------------------+
//| Calcul des valeurs de l'indicateur                               |
//+------------------------------------------------------------------+
void CandleSizeOnBuffer(const int rates_total,const int prev_calculated,
                        const double &first[],const double &second[],double &buffer[])
  {
//--- variable de départ pour le calcul des barres
```

© 2000-2024, MetaQuotes Ltd.

```
   int start=prev_calculated;
//--- travaille sur la dernière barre si les valeurs de l'indicateur ont déjà été calc
   if(prev_calculated>0)
      start--;
//--- définit le sens d'indexation des tableaux
   bool as_series_first=ArrayGetAsSeries(first);
   bool as_series_second=ArrayGetAsSeries(second);
   bool as_series_buffer=ArrayGetAsSeries(buffer);
//--- remplace le sens d'indexation avec un accès direct si nécessaire
   if(as_series_first)
      ArraySetAsSeries(first,false);
   if(as_series_second)
      ArraySetAsSeries(second,false);
   if(as_series_buffer)
      ArraySetAsSeries(buffer,false);
//--- calcule les valeurs de l'indicateur
   for(int i=start;i<rates_total;i++)
      buffer[i]=MathAbs(first[i]-second[i]);
  }
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- lie les buffers de l'indicateur
   SetIndexBuffer(0,ExtBuffer);
//--- définit le sens d'indexation du buffer de l'indicateur
   ArraySetAsSeries(ExtBuffer,InpAsSeries);
//--- vérifie pour quels prix l'indicateur est calculé
   if(InpPrices)
     {
      //--- Prix Open et Close
      PlotIndexSetString(0,PLOT_LABEL,"BodySize");
      //--- définit la couleur de l'indicateur
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrOrange);
     }
   else
     {
      //--- Prix High et Low
      PlotIndexSetString(0,PLOT_LABEL,"ShadowSize");
      //--- définit la couleur de l'indicateur
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrDodgerBlue);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- calcule l'indicateur selon la valeur du flag
   if(InpPrices)
      CandleSizeOnBuffer(rates_total,prev_calculated,open,close,ExtBuffer);
   else
      CandleSizeOnBuffer(rates_total,prev_calculated,high,low,ExtBuffer);
//--- retourne la valeur de prev_calculated pour le prochain appel
   return(rates_total);
  }
```

Voir aussi

Accès aux timeseries, ArraySetAsSeries

© 2000-2024, MetaQuotes Ltd.

ArrayInitialize

La fonction initialise un tableau numérique avec une valeur prédéfinie.

Pour l'initialisation d'un tableau de type char

```
int  ArrayInitialize(
   char    array[],     // tableau à initialiser
   char    value        // valeur à utiliser
   );
```

Pour l'initialisation d'un tableau de type short

```
int  ArrayInitialize(
   short   array[],     // tableau à initialiser
   short   value        // valeur à utiliser
   );
```

Pour l'initialisation d'un tableau de type int

```
int  ArrayInitialize(
   int     array[],     // tableau à initialiser
   int     value        // valeur à utiliser
   );
```

Pour l'initialisation d'un tableau de type long

```
int  ArrayInitialize(
   long    array[],     // tableau à initialiser
   long    value        // valeur à utiliser
   );
```

Pour l'initialisation d'un tableau de type float

```
int  ArrayInitialize(
   float   array[],     // tableau à initialiser
   float   value        // valeur à utiliser
   );
```

Pour l'initialisation d'un tableau de type double

```
int  ArrayInitialize(
   double  array[],     // tableau à initialiser
   double  value        // valeur à utiliser
   );
```

Pour l'initialisation d'un tableau de type bool

```
int  ArrayInitialize(
   bool    array[],     // tableau à initialiser
   bool    value        // valeur à utiliser
   );
```

© 2000-2024, MetaQuotes Ltd.

Pour l'initialisation d'un tableau de type uint

```
int  ArrayInitialize(
   uint    array[],     // tableau à initialiser
   uint    value        // valeur à utiliser
   );
```

Paramètres

array[]

[out] Tableau numérique à initialiser.

value

[in] Nouvelle valeur qui doit être mise pour tous les éléments du tableau.

Valeur de Retour

Nombre d'éléments initialisés.

Note

La fonction ArrayResize() permet de définir la taille d'un tableau avec une réserve pour être étendue ultérieurement sans réallocation physique de la mémoire. Elle est implémentée pour de meilleures performances, car les opérations de réallocation mémoire sont raisonnablement lentes.

L'initialisation du tableau avec ArrayInitialize(array, init_val) ne signifie pas l'initialisation de la valeur des éléments réservés alloués pour ce tableau. Lors d'une extension ultérieure du *tableau* avec la fonction ArrayResize(), les éléments sont ajoutés à la fin du tableau, leurs valeurs seront indéfinies et dans la plupart des cas ne seront pas égales à *init_value*.

Exemple :

```
void OnStart()
  {
//--- tableau dynamique
   double array[];
//--- définissons une taille de tableau pour 100 éléments et réservons un buffer pour
   ArrayResize(array,100,10);
//--- initialisons les éléments du tableau avec la valeur EMPTY_VALUE=DBL_MAX
   ArrayInitialize(array,EMPTY_VALUE);
   Print("Valeurs des 10 derniers éléments après initialisation");
   for(int i=90;i<100;i++) printf("array[%d] = %G",i,array[i]);
//--- étendons le tableau de 5 éléments
   ArrayResize(array,105);
   Print("Valeurs des 10 derniers éléments après ArrayResize(array,105)");
//--- les valeurs des 5 derniers éléments sont obtenues avec le buffer de la réserve
   for(int i=95;i<105;i++) printf("array[%d] = %G",i,array[i]);
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayFill

La fonction remplit un tableau avec une valeur spécifiée.

```
void  ArrayFill(
   void&  array[],        // tableau
   int    start,          // indice de départ
   int    count,          // nombre d'éléments à remplir
   void   value           // valeur
   );
```

Paramètres

array[]

[out] Tableau de type simple (char, uchar, short, ushort, int, uint, long, ulong, bool, color, datetime, float, double).

start

[in] Indice de départ. Dans ce cas, le flag AS_SERIES spécifié est ignoré.

count

[in] Nombre d'éléments à remplir.

value

[in] Valeur avec laquelle le tableau sera rempli.

Valeur de Retour

Aucune valeur de retour.

Note

Lorsque la fonction ArrayFill() est appelée, la direction de l'indexation (de gauche à droite) est toujours implicite. Cela signifie que le changement d'ordre d'accès aux éléments du tableau avec la fonction ArraySetAsSeries() est ignoré.

Un tableau à plusieurs dimensions est vu comme n'ayant qu'une seule dimension lorsqu'il est traité par la fonction ArrayFill(). Par exemple, array[2][4] est traité comme array[8]. Vous pouvez donc spécifier l'indice de l'élément de départ à 5 pour travailler sur ce tableau. L'appel à ArrayFill(array, 5, 2, 3.14) pour array[2][4] remplit les éléments array[1][1] et array[1][2] avec 3,14.

Exemple :

```
void OnStart()
  {
//--- déclare un tableau dynamique
   int a[];
//--- définit la taille
   ArrayResize(a,10);
//--- remplit les 5 premiers éléments avec 123
   ArrayFill(a,0,5,123);
//--- remplit les 5 éléments suivants avec 456
   ArrayFill(a,5,5,456);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- affiche les valeurs
   for(int i=0;i<ArraySize(a);i++) printf("a[%d] = %d",i,a[i]);
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayIsDynamic

La fonction vérifie si un tableau est dynamique.

```
bool  ArrayIsDynamic(
   const void&  array[]    // tableau à vérifier
   );
```

Paramètres

array[]

[in] Tableau à vérifier.

Valeur de Retour

Retourne true si le tableau sélectionné est dynamique, sinon retourne false.

Exemple :

```
#property description "Cet indicateur ne calcule pas de valeurs. Il effectue une tenta
#property description "d'appeler la fonction ArrayFree() sur 3 tableaux : dynamique, s
#property description "un buffer d'indicateur. Les résultats sont affichés dans le jou
//--- paramètres de l'indicateur
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- variables globales
double ExtDynamic[];   // tableau dynamique
double ExtStatic[100]; // tableau statique
bool   ExtFlag=true;   // flag
double ExtBuff[];      // buffer de l'indicateur
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- alloue la mémoire du tableau
   ArrayResize(ExtDynamic,100);
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,ExtBuff);
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- effectue une analyse simple
   if(ExtFlag)
     {
      //--- tentative de libérer la mémoire des tableaux
      //--- 1. Tableau dynamique
      Print("+============================+");
      Print("1. Vérification du tableau dynamique :");
      Print("Taille avant que la mémoire ne soit libérée = ",ArraySize(ExtDynamic));
      Print("Est-ce que c'est un tableau dynamique = ",ArrayIsDynamic(ExtDynamic) ? "O
      //--- tentative de libérerr la mémoire du tableau
      ArrayFree(ExtDynamic);
      Print("Taille après la libération de la mémoire = ",ArraySize(ExtDynamic));
      //--- 2. Tableau statique
      Print("2. Vérification du tableau statique :");
      Print("Taille avant que la mémoire ne soit libérée = ",ArraySize(ExtStatic));
      Print("Est-ce que c'est un tableau dynamique = ",ArrayIsDynamic(ExtStatic) ? "Ou
      //--- tentative de libérerr la mémoire du tableau
      ArrayFree(ExtStatic);
      Print("Taille après la libération de la mémoire = ",ArraySize(ExtStatic));
      //--- 3. Buffer de l'indicateur
      Print("3. Vérification du buffer de l'indicateur :");
      Print("Taille avant que la mémoire ne soit libérée = ",ArraySize(ExtBuff));
      Print("Est-ce que c'est un tableau dynamique = ",ArrayIsDynamic(ExtBuff) ? "Oui"
      //--- tentative de libérerr la mémoire du tableau
      ArrayFree(ExtBuff);
      Print("Taille après la libération de la mémoire = ",ArraySize(ExtBuff));
      //--- change la valeur du flag
      ExtFlag=false;
     }
//--- retourne la valeur de prev_calculated pour le prochain appel
   return(rates_total);
  }
```

Voir aussi

Accès aux timeseries et aux indicateurs

© 2000-2024, MetaQuotes Ltd.

ArrayIsSeries

La fonction vérifie si un tableau est une timeseries.

```
bool  ArrayIsSeries(
   const void&  array[]    // tableau à vérifier
   );
```

Paramètres

array[]

[in] Tableau à vérifier.

Valeur de Retour

Retourne true si le tableau vérifié est une timeseries, sinon retourne false. L'ordre d'accès aux éléments des tableaux passés en paramètres à la fonction OnCalculate() doit être vérifié avec ArrayGetAsSeries().

Exemple :

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- dessin de Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- buffers de l'indicateur
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'indicateur personnalisé           |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000-2024, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   if(ArrayIsSeries(open))
      Print("open[] est une timeseries");
   else
      Print("open[] n'est pas une timeseries !!!");
//--- retourne la valeur de prev_calculated pour le prochain appel
   return(rates_total);
  }
```

Voir aussi

Accès aux timeseries et aux indicateurs

© 2000-2024, MetaQuotes Ltd.

ArrayMaximum

Recherche le plus grand élément dans la première dimension d'un tableau numérique à plusieurs dimensions.

```
int  ArrayMaximum(
   const void&   array[],            // tableau où effectuer la recherche
   int           start=0,            // indice de début de recherche
   int           count=WHOLE_ARRAY   // nombre d'éléments à vérifier
   );
```

Paramètres

array[]

[in] Le tableau numérique dans lequel la recherche est faite.

start=0

[in] Indice de début de la recherche.

count=WHOLE_ARRAY

[in] Nombre d'éléments sur lesquels effectuer la recherche. Par défaut, la recherche s'effectue sur le tableau en entier (count=WHOLE_ARRAY).

Valeur de Retour

La fonction retourne l'indice de l'élément trouvé en tenant compte de la série du tableau. Encas d'échec, retourne -1.

Note

La valeur du flag AS_SERIES est prise en compte lors de la recherche du maximum.

Les fonctions ArrayMaximum et ArrayMinimum acceptent des tableaux de n'importe quelle dimension comme paramètre. Cependant, la recherche est toujours effectuée sur la première (zéro) dimension.

Exemple :

```
#property description "L'indicateur affiche les bougies d'une période supérieure sur 
//--- paramètres de l'indicateur
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- plot 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- plot 4
```

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
//---- plot 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- plot 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- plot 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- constantes prédéfinies
#define INDICATOR_EMPTY_VALUE 0.0
//--- paramètres d'entrée
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Période de calcul de l'indi
input datetime        InpDateStart=D'2013.01.01 00:00'; // Date de début de l'analyse
//--- buffers de l'indicateur pour les bougies baissières
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- buffers de l'indicateur pour les bougies haussières
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- variables globales
datetime ExtTimeBuff[];      // buffer de la période supérieure
int      ExtSize=0;          // taille du buffer
int      ExtCount=0;         // indice dans le buffer
int      ExtStartPos=0;      // position initiale pour le calcul de l'indicateur
bool     ExtStartFlag=true;  // flag supplémentaire pour conserver la position initial
datetime ExtCurrentTime[1];  // dernière heure de la génération de la barre de la pér
datetime ExtLastTime;        // dernière heure de la période supérieure pour laquelle
bool     ExtBearFlag=true;   // flag pour définir l'ordre d'écriture des données dans
bool     ExtBullFlag=true;   // flag pour définir l'ordre d'écriture des données dans
int      ExtIndexMax=0;      // indice de l'élément maximum du tableau
int      ExtIndexMin=0;      // indice de l'élément minimum du tableau
```

© 2000-2024, MetaQuotes Ltd.

```
int      ExtDirectionFlag=0; // direction du mouvement des prix pour la bougie courant
//--- décalage entre les prix open et close de la bougie pour un dessin correct
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//| Remplissage de la partie basique de la bougie                    |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- trouve l'indice des éléments maximum et minimum des tableaux
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // maximum des High
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // minimum des Low
//--- définit combien de barre de la période actuelle doivent être remplies
   int count=fill_index-start+1;
//--- si le prix de clôture de la première barre est supérieur à celui de la dernière
   if(open[start]>close[last])
     {
      //--- si la bougie était haussière avant, efface les valeurs des buffers haussie
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShac
      //--- bougie baissière
      ExtDirectionFlag=-1;
      //--- génère la bougie
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShac
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- sort de la fonction
      return;
     }
//--- si le prix de clôture de la première barre est inférieur à celui de la dernière
   if(open[start]<close[last])
     {
      //--- si la bougie était baissière avant, efface les valeurs des buffers baissie
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShac
      //--- bougie haussière
      ExtDirectionFlag=1;
      //--- génère la bougie
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShac
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- sort de la fonction
      return;
     }
//--- si vous arrivez dans cette partie de la fonction, le prix d'ouverture de la pren
//--- prix de clôture de la dernière barre ; une telle bougie est considérée comme bai
//--- si la bougie était haussière avant, efface les valeurs des buffers haussiers de
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
```

```
//--- bougie baissière
   ExtDirectionFlag=-1;
//--- si les prix de clôture et d'ouverture sont égaux, utilise le décalage pour un a
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//| Remplit la fin de la bougie                                      |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- ne dessine pas dans le cas d'une barre seule
   if(last-start==0)
      return;
//--- si le prix de clôture de la première barre est supérieur à celui de la dernière
   if(open[start]>close[last])
     {
      //--- génère la fin de la bougie
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- sort de la fonction
      return;
     }
//--- si le prix de clôture de la première barre est inférieur à celui de la dernière
   if(open[start]<close[last])
     {
      //--- génère la fin de la bougie
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- sort de la fonction
      return;
     }
//--- si vous arrivez dans cette partie de la fonction, le prix d'ouverture de la pren
//--- prix de clôture de la dernière barre ; une telle bougie est considérée comme ba
//--- génère la fin de la bougie
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEr
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vérifie la période de l'indicateur
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- affiche les prix au premier plan
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- lie les buffers de l'indicateur
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- définit certaines propriétés pour créer l'indicateur
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // type de construction grap
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // style de ligne
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // largeur de ligne
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

```
                const long &volume[],
                const int &spread[])
  {
//--- dans le cas où aucune barre n'est encore calculée
   if(prev_calculated==0)
     {
      //--- récupère l'heure d'arrivée de la barre de la plus grande période
      if(!GetTimeData())
         return(0);
     }
//--- définit l'indexation directe
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- variable de départ pour le calcul des barres
   int start=prev_calculated;
//--- si la barre est générée, recalcule la valeur de l'indicateur
   if(start!=0 && start==rates_total)
      start--;
//--- boucle de calcul des valeurs de l'indicateur
   for(int i=start;i<rates_total;i++)
     {
      //--- remplit i éléments des buffers de l'indicateur avec des valeurs vides
      FillIndicatorBuffers(i);
      //--- effectue le calcul des barres commençant à la date InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- définit la position à partir de laquelle les valeurs sont affichées pou
         if(ExtStartFlag)
           {
            //--- stocke le numéro de la barre initiale
            ExtStartPos=i;
            //--- définit la première date de la période supérieure excédant time[i]
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- change la valeur du flag afin de ne pas exécuter ce bloc à nouveau
            ExtStartFlag=false;
           }
         //--- vérifie s'il reste des éléments dans le tableau
         if(ExtCount<ExtSize)
           {
            //--- attend que la valeur time actuelle atteigne celle de la période supé
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- dessine la partie principale de la bougie (sans remplir la zone e
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,ExtI
```

```
                //--- remplit la fin de la bougie (la zone entre les deux dernières bar
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- décale la position initiale pour dessiner la prochaine bougie
                ExtStartPos=i;
                //--- augmente le compteur du tableau
                ExtCount++;
               }
            else
               continue;
            }
         else
            {
            //--- réinitialise les valeurs du tableau
            ResetLastError();
            //--- récupère la dernière date depuis la période supérieure
            if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Erreur de copie des données, code = ",GetLastError());
                return(0);
               }
            //--- si la nouvelle date est plus tard, stoppe la génération des bougies
            if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- efface la zone entre les deux dernières barres dans les buffers p
                ClearEndOfBodyMain(i-1);
                //--- remplit la zone avec les buffers supplémentaires de l'indicateur
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- décale la position initiale pour dessiner la prochaine bougie
                ExtStartPos=i;
                //--- réinitialise le flag de diraction du prix
                ExtDirectionFlag=0;
                //--- stocke la nouvelle dernière date
                ExtLastTime=ExtCurrentTime[0];
               }
            else
               {
                //--- génère la bougie
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- retourne la valeur de prev_calculated pour le prochain appel
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Vérifie la validité de la période spécifiée de l'indicateur      |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- la période de l'indicateur ne devrait pas être supérieure à la période du graphi
   if(current_period>=high_period)
     {
      Print("Erreur ! La valeur de la période de l'indicateur devrait excéder la valeu
      return(false);
     }
//--- si la période de l'indicateur est une semaine ou un mois, la période est correct
   if(high_period>32768)
      return(true);
//--- convertit les valeurs de la période en minutes
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- la période de l'indicateur devrait être un multiple de la période sur laquelle i
   if(high_period%current_period!=0)
     {
      Print("Erreur ! La valeur de la période de l'indicateur devrait être un multiple
      return(false);
     }
//--- la période de l'indicateur devrait excéder la période sur laquelle il est affich
   if(high_period/current_period<3)
     {
      Print("Erreur ! La période de l'indicateur devrait excéder la période actuelle p
      return(false);
     }
//--- la période de l'indicateur est correcte pour la période actuelle
   return(true);
  }
//+------------------------------------------------------------------+
//| Récupère les données time de la période supérieure               |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- copie toutes les données pour l'heure actuelle
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- récupère le code d'erreur
      int code=GetLastError();
      //--- affiche le message d'erreur
      PrintFormat("Erreur de copie des données ! %s",code==4401
                  ? "L'historique est toujours en cours de chargement !"
                  : "Code = "+IntegerToString(code));
      //--- retourne false pour effectuer une tentative répétée de téléchargement des
      return(false);
     }
```

```
//--- récupère la taille du tableau
   ExtSize=ArraySize(ExtTimeBuff);
//--- définit l'indice de la boucle pour le tableau à zéro
   ExtCount=0;
//--- définit la position de la bougie actuelle à zéro
   ExtStartPos=0;
   ExtStartFlag=true;
//--- stocke la dernière valeur time de la période supérieure
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------+
//| La fonction forme la partie principale de la bougie. Suivant la valeur |
//| du flag, la fonction définit quelles données et quels tableaux seront   |
//| utilisés pour un affichage correct.                                     |
//+------------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- vérifie la valeur du flag
   if(flag)
     {
      //--- génère le corps de la bougie
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- génère l'ombre de la bougie
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- génère le corps de la bougie
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- génère l'ombre de la bougie
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------+
//| La fonction forme la fin de la bougie. Suivant la valeur du flag,       |
//| la fonction définit quelles données et quels tableaux seront            |
//| utilisés pour un affichage correct.                                     |
//+------------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- vérifie la valeur du flag
   if(flag)
     {
      //--- génère la fin du corps de la bougie
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- génère la fin de l'ombre de la bougie
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- change la valeur du flag à son opposé
      flag=false;
     }
   else
     {
      //--- génère la fin du corps de la bougie
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- génère la fin de l'ombre de la bougie
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- change la valeur du flag à son opposé
      flag=true;
     }
  }
//+------------------------------------------------------------------------------+
//| Efface la fin de la bougie (la zone entre les deux dernières                 |
//| barres)                                                                      |
//+------------------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSec
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSec
  }
//+------------------------------------------------------------------------+
//| Efface la bougie                                                       |
//+------------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- vérification
   if(count!=0)
     {
      //--- remplit les buffers de l'indicateur avec des valeurs vides
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------------+
//| Génère la partie principale de la bougie                               |
//+------------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
  {
//--- vérification
   if(count!=0)
     {
      //--- remplit les buffers de l'indicateur avec les valeurs
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------------------+
//| Génère la fin de la bougie                                                   |
//+------------------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- remplit les buffers de l'indicateur avec les valeurs
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Remplit l'élément i des buffers avec des valeurs vides           |
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- définit une valeur vide dans la cellule des buffers de l'indicateur
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayMinimum

Recherche le plus petit élément dans la première dimension d'un tableau numérique à plusieurs dimensions.

```
int  ArrayMinimum(
   const void&   array[],            // tableau où effectuer la recherche
   int           start=0,            // indice de début de recherche
   int           count=WHOLE_ARRAY   // nombre d'éléments à vérifier
   );
```

Paramètres

array[]

[in] Le tableau numérique dans lequel la recherche est faite.

start=0

[in] Indice de début de la recherche.

count=WHOLE_ARRAY

[in] Nombre d'éléments sur lesquels effectuer la recherche. Par défaut, la recherche s'effectue sur le tableau en entier (count=WHOLE_ARRAY).

Valeur de Retour

La fonction retourne l'indice de l'élément trouvé en tenant compte de la série du tableau. Encas d'échec, retourne -1.

Note

La valeur du flag AS_SERIES est prise en compte lors de la recherche du minimum.

Les fonctions ArrayMaximum et ArrayMinimum acceptent des tableaux de n'importe quelle dimension comme paramètre. Cependant, la recherche est toujours effectuée sur la première (zéro) dimension.

Exemple :

```
#property description "L'indicateur affiche les bougies d'une période supérieure sur l
//--- paramètres de l'indicateur
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- plot 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- plot 4
```

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
//---- plot 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- plot 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- plot 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- constantes prédéfinies
#define INDICATOR_EMPTY_VALUE 0.0
//--- paramètres d'entrée
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Période de calcul de l'indi
input datetime        InpDateStart=D'2013.01.01 00:00'; // Date de début de l'analyse
//--- buffers de l'indicateur pour les bougies baissières
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- buffers de l'indicateur pour les bougies haussières
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- variables globales
datetime ExtTimeBuff[];      // buffer de la période supérieure
int      ExtSize=0;          // taille du buffer
int      ExtCount=0;         // indice dans le buffer
int      ExtStartPos=0;      // position initiale pour le calcul de l'indicateur
bool     ExtStartFlag=true;  // flag supplémentaire pour conserver la position initial
datetime ExtCurrentTime[1];  // dernière heure de la génération de la barre de la pér
datetime ExtLastTime;        // dernière heure de la période supérieure pour laquelle
bool     ExtBearFlag=true;   // flag pour définir l'ordre d'écriture des données dans
bool     ExtBullFlag=true;   // flag pour définir l'ordre d'écriture des données dans
int      ExtIndexMax=0;      // indice de l'élément maximum du tableau
int      ExtIndexMin=0;      // indice de l'élément minimum du tableau
```

© 2000-2024, MetaQuotes Ltd.

```
int      ExtDirectionFlag=0; // direction du mouvement des prix pour la bougie courant
//--- décalage entre les prix open et close de la bougie pour un dessin correct
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//| Remplissage de la partie basique de la bougie                    |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- trouve l'indice des éléments maximum et minimum des tableaux
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // maximum des High
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // minimum des Low
//--- définit combien de barre de la période actuelle doivent être remplies
   int count=fill_index-start+1;
//--- si le prix de clôture de la première barre est supérieur à celui de la dernière
   if(open[start]>close[last])
     {
      //--- si la bougie était haussière avant, efface les valeurs des buffers haussie
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- bougie baissière
      ExtDirectionFlag=-1;
      //--- génère la bougie
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- sort de la fonction
      return;
     }
//--- si le prix de clôture de la première barre est inférieur à celui de la dernière
   if(open[start]<close[last])
     {
      //--- si la bougie était baissière avant, efface les valeurs des buffers baissie
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- bougie haussière
      ExtDirectionFlag=1;
      //--- génère la bougie
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- sort de la fonction
      return;
     }
//--- si vous arrivez dans cette partie de la fonction, le prix d'ouverture de la pren
//--- prix de clôture de la dernière barre ; une telle bougie est considérée comme bai
//--- si la bougie était haussière avant, efface les valeurs des buffers haussiers de
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
```

```
//--- bougie baissière
   ExtDirectionFlag=-1;
//--- si les prix de clôture et d'ouverture sont égaux, utilise le décalage pour un a
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//| Remplit la fin de la bougie                                      |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- ne dessine pas dans le cas d'une barre seule
   if(last-start==0)
      return;
//--- si le prix de clôture de la première barre est supérieur à celui de la dernière
   if(open[start]>close[last])
     {
      //--- génère la fin de la bougie
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- sort de la fonction
      return;
     }
//--- si le prix de clôture de la première barre est inférieur à celui de la dernière
   if(open[start]<close[last])
     {
      //--- génère la fin de la bougie
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- sort de la fonction
      return;
     }
//--- si vous arrivez dans cette partie de la fonction, le prix d'ouverture de la pren
//--- prix de clôture de la dernière barre ; une telle bougie est considérée comme ba
//--- génère la fin de la bougie
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEr
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vérifie la période de l'indicateur
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- affiche les prix au premier plan
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- lie les buffers de l'indicateur
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- définit certaines propriétés pour créer l'indicateur
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // type de construction grap
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // style de ligne
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // largeur de ligne
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

```
                const long &volume[],
                const int &spread[])
  {
//--- dans le cas où aucune barre n'est encore calculée
   if(prev_calculated==0)
     {
      //--- récupère l'heure d'arrivée de la barre de la plus grande période
      if(!GetTimeData())
         return(0);
     }
//--- définit l'indexation directe
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- variable de départ pour le calcul des barres
   int start=prev_calculated;
//--- si la barre est générée, recalcule la valeur de l'indicateur
   if(start!=0 && start==rates_total)
      start--;
//--- boucle de calcul des valeurs de l'indicateur
   for(int i=start;i<rates_total;i++)
     {
      //--- remplit i éléments des buffers de l'indicateur avec des valeurs vides
      FillIndicatorBuffers(i);
      //--- effectue le calcul des barres commençant à la date InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- définit la position à partir de laquelle les valeurs sont affichées pou
         if(ExtStartFlag)
           {
            //--- stocke le numéro de la barre initiale
            ExtStartPos=i;
            //--- définit la première date de la période supérieure excédant time[i]
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- change la valeur du flag afin de ne pas exécuter ce bloc à nouveau
            ExtStartFlag=false;
           }
         //--- vérifie s'il reste des éléments dans le tableau
         if(ExtCount<ExtSize)
           {
            //--- attend que la valeur time actuelle atteigne celle de la période supé
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- dessine la partie principale de la bougie (sans remplir la zone e
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,ExtI
```

```
                //--- remplit la fin de la bougie (la zone entre les deux dernières bar
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- décale la position initiale pour dessiner la prochaine bougie
                ExtStartPos=i;
                //--- augmente le compteur du tableau
                ExtCount++;
               }
            else
               continue;
            }
         else
            {
            //--- réinitialise les valeurs du tableau
            ResetLastError();
            //--- récupère la dernière date depuis la période supérieure
            if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Erreur de copie des données, code = ",GetLastError());
                return(0);
               }
            //--- si la nouvelle date est plus tard, stoppe la génération des bougies
            if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- efface la zone entre les deux dernières barres dans les buffers p
                ClearEndOfBodyMain(i-1);
                //--- remplit la zone avec les buffers supplémentaires de l'indicateur
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- décale la position initiale pour dessiner la prochaine bougie
                ExtStartPos=i;
                //--- réinitialise le flag de diraction du prix
                ExtDirectionFlag=0;
                //--- stocke la nouvelle dernière date
                ExtLastTime=ExtCurrentTime[0];
               }
            else
               {
                //--- génère la bougie
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- retourne la valeur de prev_calculated pour le prochain appel
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Vérifie la validité de la période spécifiée de l'indicateur      |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- la période de l'indicateur ne devrait pas être supérieure à la période du graphi
   if(current_period>=high_period)
     {
      Print("Erreur ! La valeur de la période de l'indicateur devrait excéder la valeu
      return(false);
     }
//--- si la période de l'indicateur est une semaine ou un mois, la période est correct
   if(high_period>32768)
      return(true);
//--- convertit les valeurs de la période en minutes
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- la période de l'indicateur devrait être un multiple de la période sur laquelle i
   if(high_period%current_period!=0)
     {
      Print("Erreur ! La valeur de la période de l'indicateur devrait être un multiple
      return(false);
     }
//--- la période de l'indicateur devrait excéder la période sur laquelle il est affich
   if(high_period/current_period<3)
     {
      Print("Erreur ! La période de l'indicateur devrait excéder la période actuelle p
      return(false);
     }
//--- la période de l'indicateur est correcte pour la période actuelle
   return(true);
  }
//+------------------------------------------------------------------+
//| Récupère les données time de la période supérieure               |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- réinitialise la valeur d'erreur
   ResetLastError();
//--- copie toutes les données pour l'heure actuelle
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- récupère le code d'erreur
      int code=GetLastError();
      //--- affiche le message d'erreur
      PrintFormat("Erreur de copie des données ! %s",code==4401
                  ? "L'historique est toujours en cours de chargement !"
                  : "Code = "+IntegerToString(code));
      //--- retourne false pour effectuer une tentative répétée de téléchargement des
      return(false);
     }
```

```
//--- récupère la taille du tableau
   ExtSize=ArraySize(ExtTimeBuff);
//--- définit l'indice de la boucle pour le tableau à zéro
   ExtCount=0;
//--- définit la position de la bougie actuelle à zéro
   ExtStartPos=0;
   ExtStartFlag=true;
//--- stocke la dernière valeur time de la période supérieure
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------+
//| La fonction forme la partie principale de la bougie. Suivant la valeur |
//| du flag, la fonction définit quelles données et quels tableaux seront   |
//| utilisés pour un affichage correct.                                     |
//+------------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- vérifie la valeur du flag
   if(flag)
     {
      //--- génère le corps de la bougie
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- génère l'ombre de la bougie
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- génère le corps de la bougie
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- génère l'ombre de la bougie
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------+
//| La fonction forme la fin de la bougie. Suivant la valeur du flag,       |
//| la fonction définit quelles données et quels tableaux seront            |
//| utilisés pour un affichage correct.                                     |
//+------------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- vérifie la valeur du flag
   if(flag)
     {
      //--- génère la fin du corps de la bougie
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- génère la fin de l'ombre de la bougie
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- change la valeur du flag à son opposé
      flag=false;
     }
   else
     {
      //--- génère la fin du corps de la bougie
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- génère la fin de l'ombre de la bougie
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- change la valeur du flag à son opposé
      flag=true;
     }
  }
//+------------------------------------------------------------------------------
//| Efface la fin de la bougie (la zone entre les deux dernières
//| barres)
//+------------------------------------------------------------------------------
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSec
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSec
  }
//+------------------------------------------------------------------+
//| Efface la bougie                                                 |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- vérification
   if(count!=0)
     {
      //--- remplit les buffers de l'indicateur avec des valeurs vides
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------+
//| Génère la partie principale de la bougie                         |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
  {
//--- vérification
   if(count!=0)
     {
      //--- remplit les buffers de l'indicateur avec les valeurs
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| Génère la fin de la bougie                                       |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- remplit les buffers de l'indicateur avec les valeurs
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Remplit l'élément i des buffers avec des valeurs vides           |
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- définit une valeur vide dans la cellule des buffers de l'indicateur
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayPrint

Ecrit un tableau de types simples ou une structure simple dans le journal.

```
void  ArrayPrint(
   const void&   array[],              // tableau à écrire
   uint          digits=_Digits,       // nombre de décimales
   const string  separator=NULL,       // séparateur des valeurs des champs de la struc
   ulong         start=0,              // indice du premier élément écrit
   ulong         count=WHOLE_ARRAY,    // nombre d'éléments à écrire
   ulong         flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_
   );
```

Paramètres

array[]

[in] Tableau de type simple ou une structure simple.

digits=_Digits

[in] Le nombre de décimales pour les types réels. La valeur par défaut est _Digits.

separator=NULL

[in] Séparateur des valeurs des champs d'un élément de la structure. La valeur par défaut NULL signifie une ligne vide. Un espace est utilisé comme séparateur dans ce cas.

start=0

[in] L'indice du premier élément à écrire du tableau. Il est écrit depuis l'indice zéro par défaut.

count=WHOLE_ARRAY

[in] Nombre d'éléments du tableau à écrire. Le tableau est écrit en intégralité par défaut (count=WHOLE_ARRAY).

flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_ALIGN

[in] Combinaison de flags définissant le mode de sortie. Tous les flags sont activés par défaut :

- ARRAYPRINT_HEADER - écrit les en-têtes du tableau de structures
- ARRAYPRINT_INDEX - écrit l'indice sur le côté gauche
- ARRAYPRINT_LIMIT - n'écrit que les 100 premiers et les 100 derniers éléments. Utilisez-le si vous souhaitez n'écrire qu'une partie d'un gros tableau.
- ARRAYPRINT_ALIGN - active l'alignement des valeurs écrites - les nombres sont alignés sur la droite, les lignes sur la gauche.
- ARRAYPRINT_DATE - lors de l'écriture de type datetime, écrit les dates au format jj.mm.aaaa
- ARRAYPRINT_MINUTES - lors de l'écriture de type datetime, écrit les heures au format HH:MM
- ARRAYPRINT_SECONDS - lors de l'écriture de type datetime, écrit les heures au format HH:MM:SS

Valeur de Retour

Aucune

Note

ArrayPrint() n'écrit pas tous les champs du tableau de structures dans le journal - les champs tableaux et pointeurs d'objets ne sont pas pris en compte. Ces colonnes ne sont tout simplement pas

© 2000-2024, MetaQuotes Ltd.

écrites pour une présentation plus pratique. Si vous avez besoin d'écrire tous les champs de la structure, vous devez écrire votre propre fonction d'écriture avec le format désiré.

Exemple :

```
//--- écrit les valeurs des 10 dernières barres
   MqlRates rates[];
   if(CopyRates(_Symbol,_Period,1,10,rates))
     {
      ArrayPrint(rates);
      Print("Vérification\n[time]\t[open]\t[high]\t[low]\t[close]\t[tick_volume]\t[spr
      for(int i=0;i<10;i++)
        {
         PrintFormat("[%d]\t%s\t%G\t%G\t%G\t%G\t%G\t%G\t%I64d\t",i,
         TimeToString(rates[i].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS),
         rates[i].open,rates[i].high,rates[i].low,rates[i].close,
         rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
   else
      PrintFormat("Echec de CopyRates, code d'erreur=%d",GetLastError());
//--- exemple d'impression
/*
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295         18110       10   17
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.11930 1.12747         17829        9   15
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744         13458       10    9
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194         15362        9   12
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172         16833        9   12
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052         15933        8   10
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528         11888        9    8
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915          7284       10    5
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904          8710        9    6
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263          8956        7    7
  Vérification
   [time] [open] [high] [low] [close] [tick_volume] [spread] [real_volume]
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295 18110 10 17300175000
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.1193 1.12747 17829 9 15632176000
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744 13458 10 9593492000
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194 15362 9 12352245000
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172 16833 9 12961333000
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052 15933 8 10720384000
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528 11888 9 8084811000
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915 7284 10 5087113000
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904 8710 9 6769629000
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263 8956 7 7192138000
*/
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

FileSave, FileLoad

© 2000-2024, MetaQuotes Ltd.

ArrayRange

La fonction retourne le nombre d'éléments dans une certaine dimension du tableau.

```
int  ArrayRange(
   const void&   array[],       // tableau à utiliser
   int           rank_index     // indice de la dimension
   );
```

Paramètres

array[]

[in] Tableau à vérifier.

rank_index

[in] Indice de la dimension.

Valeur de Retour

Nombre d'éléments dans la dimension sélectionnée du tableau.

Note

Puisque les indices commencent à zéro, le nombre de dimensions du tableau est plus grand de 1 que l'indice de la dernière dimension.

Exemple :

```
void OnStart()
  {
//--- crée un tableau à 4 dimensions
   double array[][5][2][4];
//--- définit la taille de la dimension zéro
   ArrayResize(array,10,10);
//--- écrit les dimensions
   int temp;
   for(int i=0;i<4;i++)
     {
      //--- récupère la taille de la dimension i
      temp=ArrayRange(array,i);
      //--- affichage
      PrintFormat("dim = %d, range = %d",i,temp);
     }
//--- Résultat
// dim = 0, range = 10
// dim = 1, range = 5
// dim = 2, range = 2
// dim = 3, range = 4
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayResize

La fonction définit une nouvelle taille pour la première dimension

```
int  ArrayResize(
   void&  array[],              // tableau passé par référence
   int    new_size,             // nouvelle taille du tableau
   int    reserve_size=0        // valeur de la taille de réserve (excès)
   );
```

Paramètres

array[]

[out] Tableau à redimmensionner.

new_size

[in] Nouvelle taille pour la première dimension.

reserve_size=0

[in] Taille distribuée à réserver.

Valeur de Retour

En cas d'exécution réussie, retourne le nombre d'éléments contenus dans le tableau après la redimmensionnement, sinon retourne -1 et le tableau n'est pas redimensionné.

Si ArrayResize() est appliqué à un tableau statique, une timeseries ou le buffer d'un indicateur, la taille du tableau reste la même - ce tableaux ne peuvent pas être réalloués. Dans ce cas, si *new_size*<=ArraySize(*array*), la fonction ne retourne que *new_size* ; sinon elle retourne -1.

Note

La fonction ne peut être appliquée qu'à des tableaux dynamiques. Il est à noter que vous ne pouvez pas changer la taille des tableaux dynamiques assignés aux buffers d'un indicateur par la fonction SetIndexBuffer(). Pour les buffers d'indicateur, toutes les opérations de redimmensionnement sont effectuées par le système d'exécution du terminal.

Le nombre total d'éléments du tableau ne peut pas excéder 2 147 483 647.

En cas d'allocation mémoire fréquente, il est recommandé d'utiliser un troisième paramètre pour définir une réserve afin de réduire le nombre d'allocations mémoire physiques. Tous les appels ultérieurs à ArrayResize ne mènent pas à une réallocation physique de la mémoire, mais change uniquement la taille de la première dimension du tableau dans la mémoire réservée. Le troisième paramètre ne sera utilisé qu'en cas d'allocation physique de la mémoire. Par exemple :

```
ArrayResize(arr,1000,1000);
for(int i=1;i<3000;i++)
   ArrayResize(arr,i,1000);
```

Dans ce cas, la mémoire sera réallouée deux foix, d'abord avant d'entrer dans la boucle de 3000 éléments (la taille du tableau sera définie à 1000), et la seconde fois lorsque i sera égal à 2000. Si nous oublions le troisième paramètre, il y a aura 2000 réallocations physiques de la mémoire, ce qui ralentira le programme.

© 2000-2024, MetaQuotes Ltd.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Compteurs
   ulong start=GetTickCount();
   ulong now;
   int   count=0;
//--- Tableau pour la démonstration de la version rapide
   double arr[];
   ArrayResize(arr,100000,100000);
//--- Vérifie la vitesse de la version avec réservation de la mémoire
   Print("--- Test Rapide: ArrayResize(arr,100000,100000)");
   for(int i=1;i<=300000;i++)
     {
      //--- Définit une nouvelle taille de tableau en spécifiant une réserve de 100 00
      ArrayResize(arr,i,100000);
      //--- Lorsqu'un chiffre rond est atteint, affiche la taille du tableau et le ter
      if(ArraySize(arr)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(arr)=%d Time=%d ms",count,ArraySize(arr),(now-star
         start=now;
        }
     }
//--- Affiche maintenant la vitesse de la version lente sans réservation de la mémoire
   double slow[];
   ArrayResize(slow,100000,100000);
//---
   count=0;
   start=GetTickCount();
   Print("---- Test Lent: ArrayResize(slow,100000)");
//---
   for(int i=1;i<=300000;i++)
     {
      //--- Définit une nouvelle taille de tableau mais sans spécifier une réserve sup
      ArrayResize(slow,i);
      //--- Lorsqu'un chiffre rond est atteint, affiche la taille du tableau et le ter
      if(ArraySize(slow)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(slow)=%d Temps passé=%d ms",count,ArraySize(slow),
         start=now;
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      }
   }
//--- Un exemple de résultat du script
/*
   Test_ArrayResize (EURUSD,H1)   --- Test Rapide : ArrayResize(arr,100000,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(arr)=100000 Temps passé=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(arr)=200000 Temps passé=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(arr)=300000 Temps passé=0 ms
   Test_ArrayResize (EURUSD,H1)   ---- Test Lent : ArrayResize(slow,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(slow)=100000 Temps passé=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(slow)=200000 Temps passé=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(slow)=300000 Temps passé=228511 ms
*/
```

Voir aussi

ArrayInitialize

© 2000-2024, MetaQuotes Ltd.

ArrayInsert

Insère le nombre spécifié d'éléments d'un tableau source dans un tableau destination à partir d'un indice spécifié.

```
bool  ArrayInsert(
   void&        dst_array[],          // tableau destination
   const void&  src_array[],          // tableau source
   uint         dst_start,            // indice de début d'insertion dans le tableau d
   uint         src_start=0,          // indice du tableau source
   uint         count=WHOLE_ARRAY     // nombre d'éléments à insérer
   );
```

Parameters

dst_array[]

[in][out] Tableau destination dans lequel les éléments doivent être ajoutés.

src_array[]

[in] Tableau source depuis lequel les éléments seront ajoutés.

dst_start

[in] Indice dans le tableau destination pour insérer les éléments depuis le tableau source.

src_start=0

[in] Indice dans le tableau source, à partir duquel les éléments du tableau source sont pris pour l'insertion.

count

[in] Nombre d'éléments à ajouter depuis le tableau source. WHOLE_ARRAY signifie tous les éléments à partir de l'indice spécifié jusqu'à la fin du tableau.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir des informations sur l'erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 5052 - ERR_SMALL_ARRAY (le paramètre *start* et/ou *count* sont incorrects ou le tableau source *src_array[]* est vide),
- 5056 - ERR_SERIES_ARRAY (le tableau ne peut pas être changé, buffer de l'indicateur),
- 4006 - ERR_INVALID_ARRAY (la copie sur le même tableau n'est pas autorisée, ou les tableaux sont de types différents, ou l'un des tableaux de taille fixe contient des objets de classe ou des structures),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (le tableau ne contient aucune structure POD, signifiant qu'une copie simple n'est pas possible),
- Les erreurs survenues lors du changement de la taille du tableau destination *dst_array[]* sont fournies dans la description de la fonction ArrayRemove().

Note

Si la fonction est utilisée pour un tableau de taille fixe, la taille du tableau destination *dst_array[]* lui-même ne change pas. A partir de la position *dst_start*, les éléments du tableau destination sont

© 2000-2024, MetaQuotes Ltd.

décalés vers la droite (les derniers *comptes* des éléments "décollent"), tandis que les éléments copiés depuis le tableau source prennent leurs places.

Vous ne pouvez pas insérer les éléments vers un tableau dynamique désigné comme buffer d'un indicateur avec la fonction SetIndexBuffer(). Pour les buffers d'indicateurs, toutes les opérations de changement de taille sont effectuées par le sous-système d'exécution du terminal.

Dans le tableau source, les éléments sont copiés depuis l'indice *src_start*. La taille du tableau source reste inchangée. Les éléments devant être ajoutés dans le tableau destination ne sont pas liés aux éléments du tableau source. Cela signifie que les changements suivants des éléments dans n'importe lequel des deux tableaux ne seront pas reflétés dans le second tableau.

Exemple:

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare le tableau de taille fixe et remplit les valeurs
   int array_dest[10];
   for(int i=0;i<10;i++)
     {
      array_dest[i]=i;
     }
   //--- tableau source
   int array_source[10];
   for(int i=0;i<10;i++)
     {
      array_source[i]=10+i;
     }
//--- affiche les tableaux avant d'insérer les éléments
   Print("Avant d'appeler ArrayInsert()");
   ArrayPrint(array_dest);
   ArrayPrint(array_source);
//--- insère 3 éléments du tableau source et affiche le nouvel ensemble du tableau des
   ArrayInsert(array_dest,array_source,4,0,3);
   Print("Après l'appel à ArrayInsert()");
   ArrayPrint(array_dest);
/*
  Résultat d'exécution
   Avant d'appeler ArrayInsert()
   0 1 2 3 4 5 6 7 8 9
   Après l'appel à ArrayInsert()
   0 1 2 3 10 11 12 7 8 9
*/
```

Voir également

ArrayRemove, ArrayCopy, ArrayResize, ArrayFree

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

ArrayRemove

Supprime le nombre spécifié d'éléments du tableau à partir de l'indice spécifié.

```
bool  ArrayRemove(
   void&        array[],            // tableau de n'importe quel type
   uint         start,              // indice à partir duquel la suppression s'effectu
   uint         count=WHOLE_ARRAY   // nombre d'éléments
   );
```

Parameters

array[]

[in][out] Tableau.

start

[in] Indice à partir duquel les éléments du tableau sont supprimés.

count=WHOLE_ARRAY

[in] Nombre d'éléments à supprimer. WHOLE_ARRAY signifie tous les éléments à partir de l'indice spécifié jusqu'à la fin du tableau.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir des informations sur l'erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 5052 - ERR_SMALL_ARRAY (valeur *start* trop grande),
- 5056 - ERR_SERIES_ARRAY (le tableau ne peut pas être changé, buffer de l'indicateur),
- 4003 - ERR_INVALID_PARAMETER (valeur *count* trop grande),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (tableau de taille fixe contenant des objets complexes avec un destructeur),
- 4006 - ERR_INVALID_ARRAY (tableau de taille fixe contenant des classes ou des structures avec un destructeur).

Note

Si la fonction est utilisée pour un tableau de taille fixe, la taille du tableau ne change pas : la "fin" restante est physiquement copiée à la position *start*. Pour une compréhension précise du fonctionnement de la fonction, reportez-vous à l'exemple ci-dessous. La copie "physique" signifie que les objets copiés ne sont pas créés en appelant le constructeur ou l'opérateur de copie. Au lieu de cela, la représentation binaire d'un objet est copiée. Pour cette raison, vous ne pouvez pas appliquer la fonction ArrayRemove() au tableau de taille fixe contenant des objets avec un destructeur (l'erreur ERR_INVALID_ARRAY ou ERR_STRUCT_WITHOBJECTS_ORCLASS est activée). Lors de la suppression de ce type d'objet, le destructeur doit être appelé 2 fois - pour l'objet d'origine et pour sa copie.

Vous ne pouvez pas supprimer les éléments de tableaux dynamiques désignés comme buffers d'un indicateur avec la fonction SetIndexBuffer(). Cela résultera en l'erreur ERR_SERIES_ARRAY. Pour les buffers d'indicateurs, toutes les opérations de changement de taille sont effectuées par le sous-système d'exécution du terminal.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare le tableau de taille fixe et remplit les valeurs
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- affiche le tableau avant de supprimer les éléments
   Print("Avant d'appeler ArrayRemove()");
   ArrayPrint(array);
//--- supprime 2 éléments du tableau et affiche le nouvel ensemble
   ArrayRemove(array,4,2);
   Print("Après l'appel à ArrayRemove()");
   ArrayPrint(array);
/*
  Résultat d'exécution :
  Avant d'appeler ArrayRemove()
   0 1 2 3 4 5 6 7 8 9
  Après l'appel à ArrayRemove()
   0 1 2 3 6 7 8 9 8 9
*/
```

Voir également

ArrayInsert, ArrayCopy, ArrayResize, ArrayFree

© 2000-2024, MetaQuotes Ltd.

ArrayReverse

Inverse le nombre spécifié d'éléments dans le tableau en commencant à l'indice spécifié.

```
bool  ArrayReverse(
   void&        array[],            // tableau de n'importe quel type
   uint         start=0,            // indice de début d'inversion
   uint         count=WHOLE_ARRAY   // nombre d'éléments
   );
```

Parameters

array[]

[in][out] Tableau.

start=0

[in] Indice de début d'inversion du tableau.

count=WHOLE_ARRAY

[in] Nombre d'éléments inversés. Si WHOLE_ARRAY, alors tous les éléments du tableau sont déplacés de manière inversée en commencant à l'indice *start* spécifié et jusqu'à la fin du tableau.

Valeur de Retour

Retourne true en cas de succès, sinon false.

Note

La fonction ArraySetAsSeries() ne déplace pas physiquement les éléments du tableau. Au lieu de cela, elle ne change que la direction d'indexation pour arranger l'accès aux éléments comme dans une timeserie. La fonction ArrayReverse() déplace physiquement les éléments du tableau afin que le tableau soit "inversé".

Exemple:

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclare le tableau de taille fixe et remplit les valeurs
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- affiche le tableau avant d'inverser les éléments
   Print("Avant d'appeler ArrayReverse()");
   ArrayPrint(array);
//--- inverse 3 éléments du tableau et affiche le nouvel ensemble
   ArrayReverse(array,4,3);
   Print("Après l'appel à ArrayReverse()");
```

© 2000-2024, MetaQuotes Ltd.

```
   ArrayPrint(array);
/*
  Résultat d'exécution :
  Avant d'appeler ArrayReverse()
   0 1 2 3 4 5 6 7 8 9
  Après l'appel à ArrayReverse()
   0 1 2 3 6 5 4 7 8 9
*/
```

Voir également

ArrayInsert, ArrayRemove, ArrayCopy, ArrayResize, ArrayFree, ArrayGetAsSeries, ArraySetAsSeries

© 2000-2024, MetaQuotes Ltd.

ArraySetAsSeries

La fonction définit le flag AS_SERIES d'un tableau dynamique, et ses éléments seront indexés comme pour les timeseries.

```
bool  ArraySetAsSeries(
   const void&  array[],     // tableau par référence
   bool         flag         // true signifie l'ordre inverse d'indexation
   );
```

Paramètres

array[]

[in][out] Tableau numérique à modifier.

flag

[in] Direction d'indexation du tableau.

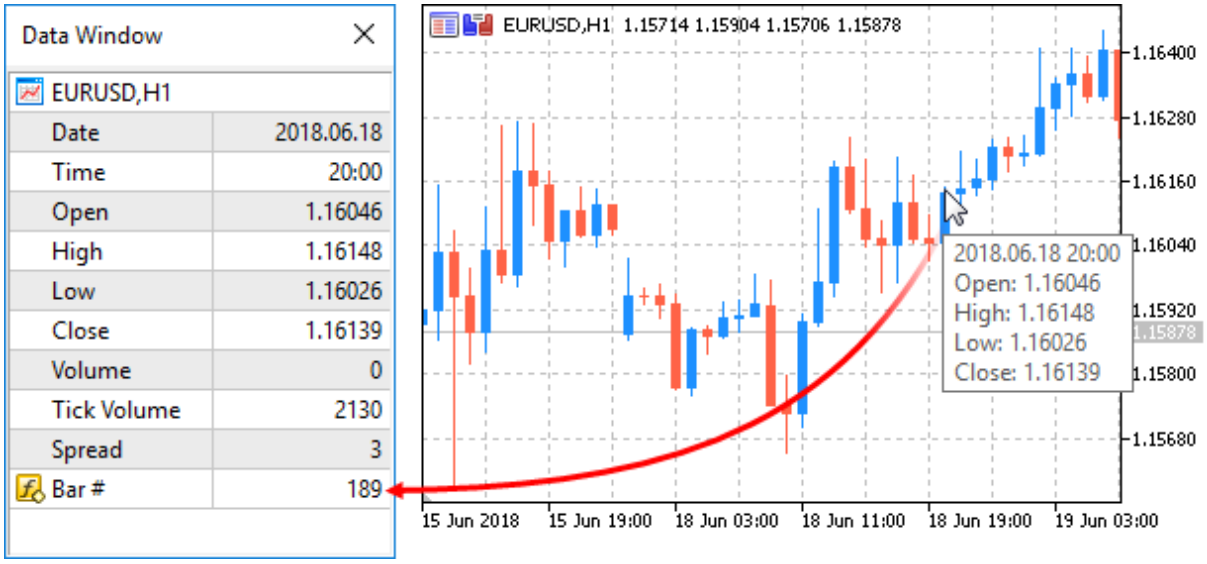
Valeur de Retour

La fonction retourne true en cas de succès, sinon - false.

Note

Le flag AS_SERIES ne peut pas être défini pour les tableaux à plusieurs dimensions ou pour les tableaux statiques (tableaux dont la taille entre crochets est définit au moment de la compilation). L'indexation des timeseries diffère de celle d'un tableau classique car les éléments des timeseries sont indexés depuis la fin vers le début (du plus récent au plus ancien).

Exemple : Indicateur affichant le numéro des barres




```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- dessin des numéros
```

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_label1  "Numeration"
#property indicator_type1   DRAW_LINE
#property indicator_color1  CLR_NONE
//--- buffers de l'indicateur
double         NumerationBuffer[];
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- définit l'indexation du buffer comme les timeseries
   ArraySetAsSeries(NumerationBuffer,true);
//--- définit la précision d'affichage dans DataWindow
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- comment le nom du tableau de l'indicateur est affiché dans DataWindow
   PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---  nous stockerons l'heure d'ouverture de la barre zéro courante
   static datetime currentBarTimeOpen=0;
//--- renverse l'accès au tableau time[] - mettons le comme dans les timeseries
   ArraySetAsSeries(time,true);
//--- Si l'heure de la barre zéro diffère de celle stockée
   if(currentBarTimeOpen!=time[0])
     {
     //--- énumère toutes les barres depuis la barre courante jusqu'au début du graphi
      for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
      currentBarTimeOpen=time[0];
     }
//--- retourne la valeur de prev_calculated pour le prochain appel
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

Voir aussi

Accès aux timeseries, ArrayGetAsSeries

© 2000-2024, MetaQuotes Ltd.

ArraySize

La fonction retourne le nombre d'éléments d'un tableau sélectionné.

```
int  ArraySize(
   const void&  array[]    // tableau à vérifier
   );
```

Paramètres

array[]

[in] Tableau de n'importe quel type.

Valeur de Retour

Valeur du type int.

Note

Pour un tableau à 1 dimension, la valeur retournée par ArraySize est égale à celle de ArrayRange(array,0).

Exemple :

```
void OnStart()
  {
//--- crée les tableaux
   double one_dim[];
   double four_dim[][10][5][2];
//--- tailles
   int one_dim_size=25;
   int reserve=20;
   int four_dim_size=5;
//--- variable supplémentaire
   int size;
//--- alloue la mémoire sans réserve
   ArrayResize(one_dim,one_dim_size);
   ArrayResize(four_dim,four_dim_size);
//--- 1. tableau à 1 dimension
   Print("+==============================================================+");
   Print("Tailles des tableaux :");
   Print("1. Tableau à 1 dimension");
   size=ArraySize(one_dim);
   PrintFormat("Taille de la dimension Zéro = %d, Taille du tableau = %d",one_dim_size
//--- 2. Tableau à plusieurs dimensions
   Print("2. Tableau à plusieurs dimensions");
   size=ArraySize(four_dim);
   PrintFormat("Taille de la dimension Zéro = %d, Taille du tableau = %d",four_dim_siz
//--- tailles des dimensions
   int d_1=ArrayRange(four_dim,1);
   int d_2=ArrayRange(four_dim,2);
   int d_3=ArrayRange(four_dim,3);
```

© 2000-2024, MetaQuotes Ltd.

```
   Print("Vérification :");
   Print("Dimension zéro = Taille du tableau / (Première dimension * Deuxième dimensio
   PrintFormat("%d = %d / (%d * %d * %d)",size/(d_1*d_2*d_3),size,d_1,d_2,d_3);
//--- 3. tableau à 1 dimension avec réserve de mémoire
   Print("3. Tableau à 1 dimension avec réserve de mémoire");
//--- double la valeur
   one_dim_size*=2;
//--- allocatation mémoire avec réserve
   ArrayResize(one_dim,one_dim_size,reserve);
//--- affiche la taille
   size=ArraySize(one_dim);
   PrintFormat("Taille avec réserve = %d, Taille réelle du tableau = %d",one_dim_size-
  }
```

© 2000-2024, MetaQuotes Ltd.

ArraySort

Trie les valeurs de la première dimension d'un tableau numérique à plusieurs dimensions par ordre croissant.

```
bool  ArraySort(
   void&  array[]      // tableau à trier
   );
```

Paramètres

array[]

[in][out] Tableau numérique à trier.

Valeur de Retour

La fonction retourne true en cas de succès, sinon - false.

Note

Un tableau est toujours trié par ordre croissant, indépendamment de la valeur du flag AS_SERIES.

Les fonctions ArraySort et ArrayBSearch acceptent des tableaux de n'importe quelle dimension comme paramètre. Cependant, la recherche et le tri sont toujours appliqués à la première (zéro) dimension.

Exemple :

```
#property description "L'indicateur analyse les données du dernier mois et dessine tou
#property description "et de gros volumes de ticks. Le tableau du volume des ticks est
#property description "pour déterminer ces bougies. Les bougies ayant un volume inféri
#property description "InpSmallVolume du tableau sont considérés comme faibles. Les bo
#property description "pourcent de InpBigVolume du tableau sont considérés comme gros.
//--- paramètres de l'indicateur
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot
#property indicator_label1  "VolumeFactor"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrDodgerBlue,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- constantes prédéfinies
#define INDICATOR_EMPTY_VALUE 0.0
//--- paramètres d'entrée
input int InpSmallVolume=15; // Valeur en pourcentage des volumes faibles (<50)
input int InpBigVolume=20;   // Valeur en pourcentage des gros volumes (<50)
//--- heure de début d'analyse (sera décalée)
datetime ExtStartTime;
//--- buffers de l'indicateur
double   ExtOpenBuff[];
```

© 2000-2024, MetaQuotes Ltd.

```
double   ExtHighBuff[];
double   ExtLowBuff[];
double   ExtCloseBuff[];
double   ExtColorBuff[];
//--- valeurs limites de volume pour l'affichage des bougies
long     ExtLeftBorder=0;
long     ExtRightBorder=0;
//+------------------------------------------------------------------+
//| Retourne les volumes des bordures pour les volumes de ticks      |
//+------------------------------------------------------------------+
bool GetVolumeBorders(void)
  {
//--- variables
   datetime stop_time;  // copie de l'heure de fin
   long     buff[];     // buffer pour la copie
//--- l'heure de fin est l'heure actuelle
   stop_time=TimeCurrent();
//--- l'heure de début est 1 mois plus tôt que l'heure actuelle
   ExtStartTime=GetStartTime(stop_time);
//--- récupère les valeurs des volumes des ticks
   ResetLastError();
   if(CopyTickVolume(Symbol(),Period(),ExtStartTime,stop_time,buff)==-1)
     {
      //--- échec de récupération des données, retourne false pour lancer la commande
      PrintFormat("Echec de récupération des valeurs de volume des ticks. Code d'erreu
      return(false);
     }
//--- calcule la taille du tableau
   int size=ArraySize(buff);
//--- trie le tableau
   ArraySort(buff);
//--- définit les valeurs des bordures gauches et droites pour les volumes des ticks
   ExtLeftBorder=buff[size*InpSmallVolume/100];
   ExtRightBorder=buff[(size-1)*(100-InpBigVolume)/100];
//--- exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Récupère les données d'un mois auparavant                        |
//+------------------------------------------------------------------+
datetime GetStartTime(const datetime stop_time)
  {
//--- convertit l'heure de fin en une variable de type MqlDateTime
   MqlDateTime temp;
   TimeToStruct(stop_time,temp);
//--- récupère les données d'un mois auparavant
   if(temp.mon>1)
      temp.mon-=1;  // le mois courant n'est pas le premier de l'année, donc le numéro
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       temp.mon=12;  // le mois courant est le premier de l'année, donc le numéro du mo
       temp.year-=1; // et le numéro de l'année vaut 1 de moins
      }
//--- le numéro du jour n'excédera pas 28
   if(temp.day>28)
       temp.day=28;
//--- retourne la date obtenue
   return(StructToTime(temp));
  }
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vérifie si les paramètres d'entrée satisfont les conditions
   if(InpSmallVolume<0 || InpSmallVolume>=50 || InpBigVolume<0 || InpBigVolume>=50)
      {
       Print("Paramètres d'entrée incorrects");
       return(INIT_PARAMETERS_INCORRECT);
      }
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,ExtOpenBuff);
   SetIndexBuffer(1,ExtHighBuff);
   SetIndexBuffer(2,ExtLowBuff);
   SetIndexBuffer(3,ExtCloseBuff);
   SetIndexBuffer(4,ExtColorBuff,INDICATOR_COLOR_INDEX);
//--- définit la valeur qui ne sera pas affichée
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- définit les étiquettes pour les buffers de l'indicateur
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- vérifie s'il y a toujours des barres non gérées
```

© 2000-2024, MetaQuotes Ltd.

```
   if(prev_calculated<rates_total)
     {
      //--- récupère les nouvelles valeurs des bordures droite et gauche pour les volu
      if(!GetVolumeBorders())
         return(0);
     }
//--- variable de départ pour le calcul de la barre
   int start=prev_calculated;
//--- travaille sur la dernière barre si les valeurs de l'indicateur ont déjà été calc
   if(start>0)
      start--;
//--- définit l'indexation directe des timeseries
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
   ArraySetAsSeries(tick_volume,false);
//--- boucle de calcul des valeurs de l'indicateur
   for(int i=start;i<rates_total;i++)
     {
      //--- remplit les bougies commençant à la date initiale
      if(ExtStartTime<=time[i])
        {
         //--- si la valeur n'est pas inférieure à la bordure droite, remplit la bougi
         if(tick_volume[i]>=ExtRightBorder)
           {
            //--- détermine les données pour dessiner la bougie
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- DodgerBlue color
            ExtColorBuff[i]=0;
            //--- continue la boucle
            continue;
           }
         //--- remplit les bougies si les valeurs ne sont pas supérieures à la bordure
         if(tick_volume[i]<=ExtLeftBorder)
           {
            //--- détermine les données pour dessiner la bougie
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- couleur orange
            ExtColorBuff[i]=1;
            //--- continue la boucle
            continue;
```

© 2000-2024, MetaQuotes Ltd.

```
            }
         }
      //--- définit les valeurs vides des barres n'ayant pas été incluses dans le calc
      ExtOpenBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtHighBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtLowBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtCloseBuff[i]=INDICATOR_EMPTY_VALUE;
     }
//--- retourne la valeur de prev_calculated pour le prochain appel
   return(rates_total);
  }
```

Voir aussi

ArrayBsearch

© 2000-2024, MetaQuotes Ltd.

ArraySwap

Intervertit les contenus de deux tableaux dynamiques du même type. Pour les tableaux à plusieurs dimensions, le nombre d'éléments de toutes les dimensions, exceptée la première, doit correspondre.

```
bool  ArraySwap(
   void&  array1[],      // premier tableau
   void&  array2[]       // deuxième tableau
   );
```

Paramètres

array1[]

[in][out] Tableau de type numérique.

array2[]

[in][out] Tableau de type numérique.

Valeur de Retour

Retourne true en cas de succès, sinon false. Dans ce cas, GetLastError() retourne le code d'erreur ERR_INVALID_ARRAY.

Note

La fonction accepte des tableaux dynamiques du même type et de même dimensions, exceptée pour la première. Pour les types integer, le signe est ignoré, c'est à dire que char==uchar

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tableaux pour stocker les cotations
   double source_array[][8];
   double   dest_array[][8];
   MqlRates rates[];
//--- récupère les données des 20 dernières bougies de la période courante
   int copied=CopyRates(NULL,0,0,20,rates);
   if(copied<=0)
     {
      PrintFormat("Echec de CopyRates(%s,0,0,20,rates), erreur=%d",
                  Symbol(),GetLastError());
      return;
     }
//--- définit la taille du tableau pour correspondre à la quantité de données copiées
   ArrayResize(source_array,copied);
//--- remplit le tableau rate_array_1[] avec les données de rates[]
   for(int i=0;i<copied;i++)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      source_array[i][0]=(double)rates[i].time;
      source_array[i][1]=rates[i].open;
      source_array[i][2]=rates[i].high;
      source_array[i][3]=rates[i].low;
      source_array[i][4]=rates[i].close;
      source_array[i][5]=(double)rates[i].tick_volume;
      source_array[i][6]=(double)rates[i].spread;
      source_array[i][7]=(double)rates[i].real_volume;
     }
//--- intervertit les données entre les tableaux source_array[] et dest_array[]
   if(!ArraySwap(source_array,dest_array))
     {
      PrintFormat("Echec de ArraySwap(source_array,rate_array_2), code d'erreur=%d",Ge
      return;
     }
//--- vérifie que le tableau source est à zéro après l'interversion
   PrintFormat("ArraySwap() effectué : ArraySize(source_array)=%d",ArraySize(source_ar
//--- affiche les données du tableau destination dest_array[]
   ArrayPrint(dest_array);
  }
```

Voir aussi

ArrayCopy, ArrayFill, ArrayRange, ArrayIsDynamic

© 2000-2024, MetaQuotes Ltd.

Matrices et vecteurs

Une matrice est un tableau à 2 dimensions d'éléments doubles, flottants ou complexes.

Un vecteur est un tableau à 1 dimension d'éléments doubles, flottants ou complexes. Le vecteur n'a aucune indication sur le fait qu'il soit vertical ou horizontal. Cela est déterminé à partir du contexte d'utilisation. Par exemple, l'utilisation du vecteur Dot assume que le vecteur de gauche est horizontal et que le vecteur de droite est vertical. Si l'indication du type est requise, une matrice à 1 ligne ou à 1 colonne peut être utilisée. Cependant, ce n'est généralement pas nécessaire.

Les matrices et les vecteurs allouent dynamiquement de la mémoire pour les données. En fait, les matrices et les vecteurs sont des objets qui ont certaines propriétés, telles que le type de données qu'ils contiennent et les dimensions. Les propriétés de matrice et de vecteur peuvent être obtenues à l'aide de méthodes telles que vector_a.Size(), matrix_b.Rows(), vector_c.Norm(), matrix_d.Cond() et autres. Toutes les dimensions peuvent être modifiées.

Lors de la création et de l'initialisation des matrices, des méthodes dites statiques sont utilisées (elles sont comme les méthodes statiques d'une classe). Par exemple : matrix::Eye(), matrix::Identity(), matrix::Ones(), vector::Ones(), matrix::Zeros(), vector::Zeros(), matrix::Full( ), vector::Full(), matrix::Tri().

Pour le moment, les opérations matricielles et vectorielles n'impliquent pas l'utilisation du type de données complexe, car cette direction de développement n'est pas encore terminée.

MQL5 prend en charge le passage de matrices et de vecteurs aux DLL. Cela permet l'importation de fonctions utilisant les types pertinents, à partir de variables externes.

Les matrices et les vecteurs sont transmis à une DLL en tant que pointeur vers un buffer. Par exemple, pour passer une matrice de type float, le paramètre correspondant de la fonction exportée depuis la DLL doit prendre un pointeur de buffer de type float.

MQL5

```
#import "mmlib.dll"
bool sgemm(uint flags, matrix<float> &C, const matrix<float> &A, const matrix<float> &
#import
```

C++

```
extern "C" __declspec(dllexport) bool sgemm(UINT flags, float *C, const float *A, cons
```

En plus des buffers, vous devez transmettre des tailles de matrice et de vecteur pour un traitement correct.

Toutes les méthodes matricielles et vectorielles sont répertoriées ci-dessous par ordre alphabétique.

Fonction	Action	Catégorie
Activation	Calcule les valeurs de la fonction d'activation et les écrit dans le vecteur/matrice passé	Machine learning

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Catégorie
ArgMax	Renvoie l'indice de la valeur maximale	Statistiques
ArgMin	Renvoie l'indice de la valeur minimale	Statistiques
ArgSort	Renvoie l'index trié	Manipulations
Assign	Copie une matrice, un vecteur ou un tableau avec une distribution automatique	Initialisation
Average	Calcule la moyenne pondérée des valeurs de matrice/vecteur	Statistiques
Cholesky	Calcule la décomposition de Cholesky	Transformations
Clip	Limite les éléments d'une matrice/vecteur à une plage donnée de valeurs valides	Manipulations
Col	Renvoie un vecteur colonne. Ecrit un vecteur dans la colonne spécifiée.	Manipulations
Cols	Retourne le nombre de colonnes d'une matrice	Fonctionnalités
Compare	Compare les éléments de deux matrices/vecteurs avec la précision spécifiée	Manipulations
CompareByDigits	Compare les éléments de deux matrices/vecteurs avec la précision des chiffres significatifs	Manipulations
Cond	Calcule le conditionnement d'une matrice	Fonctionnalités
Convolve	Renvoie la convolution discrète et linéaire de deux vecteurs	Opérations
Copy	Renvoie une copie de la matrice/du vecteur donné	Manipulations
CopyIndicatorBuffer	Retourne la quantité spécifiée de données du buffer spécifié de l'indicateur vers un vecteur	Initialisation
CopyRates	Récupère la série historique de la structure MqlRates pour le symbole et la période spécifiés,	Initialisation

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Catégorie
	avec le nombre spécifié d'éléments dans une matrice ou un vecteur	
CopyTicks	Récupère les ticks d'une structure MqlTick dans une matrice ou un vecteur	Initialisation
CopyTicksRange	Récupère les ticks d'une structure MqlTick dans une matrice ou un vecteur dans la plage de dates spécifiée	Initialisation
CorrCoef	Calcul du coefficient de corrélation de Pearson (coefficient de corrélation linéaire)	Opérations
Correlate	Calcul de la corrélation croisée de deux vecteurs	Opérations
Cov	Calcul de la matrice de covariance	Opérations
CumProd	Renvoie le produit cumulé des éléments de matrice/vecteur, y compris ceux le long de l'axe donné	Statistiques
CumSum	Renvoie la somme cumulée des éléments de matrice/vecteur, y compris ceux le long de l'axe donné	Statistiques
Derivative	Calcule les valeurs dérivées de la fonction d'activation et les écrit dans le vecteur/matrice passé	Machine learning
Det	Calcule le déterminant d'une matrice carrée inversible	Fonctionnalités
Diag	Extrait une diagonale ou construit une matrice diagonale	Manipulations
Dot	Produit scalaire de deux vecteurs	Opérations
Eig	Calcule les valeurs propres et les vecteurs propres droits d'une matrice carrée	Transformations
EigVals	Calcule les valeurs propres d'une matrice générale	Transformations

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Catégorie
Eye	Renvoie une matrice avec des 1 sur la diagonale et des 0 ailleurs	Initialisation
Fill	Remplit une matrice ou un vecteur existant avec la valeur spécifiée	Initialisation
Flat	Accès à un élément de la matrice via un index au lieu de deux	Manipulations
Full	Crée et retourne une nouvelle matrice remplie avec la valeur donnée	Initialisation
GeMM	La méthode GeMM (General Matrix Multiply) implémente la multiplication générale de deux matrices	Opérations
HasNan	Retourne le nombre de valeurs NaN dans une matrice/un vecteur	Manipulations
Hsplit	Divise une matrice horizontalement en plusieurs sous-matrices. Similaire à Split avec axis=0	Manipulations
Identity	Crée une matrice d'identité de la taille spécifiée	Initialisation
Init	Initialisation matricielle ou vectorielle	Initialisation
Inner	Produit interne de deux matrices	Opérations
Inv	Calcule l'inverse multiplicatif d'une matrice carrée inversible par la méthode de Jordan-Gauss	Solutions
Kron	Calcul du produit de Kronecker de deux matrices, matrice et vecteur, vecteur et matrice ou deux vecteurs	Opérations
LinearRegression	Calcule un vecteur/une matrice avec des valeurs de régression linéaire calculées	Statistiques
Loss	Calcule les valeurs de la fonction de perte et les écrit dans le vecteur/matrice passé	Machine learning

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Catégorie
LstSq	Renvoie la solution des moindres carrés des équations algébriques linéaires (pour les matrices non carrées ou dégénérées)	Solutions
LU	Implémente une décomposition LU d'une matrice : le produit d'une matrice triangulaire inférieure et d'une matrice triangulaire supérieure	Transformations
LUP	Implémente une factorisation LUP avec permutation partielle, qui fait référence à la décomposition LU avec permutations de lignes uniquement : PA=LU	Transformations
MatMul	Produit matriciel de deux matrices	Opérations
Max	Renvoie la valeur maximale dans une matrice/vecteur	Statistiques
Mean	Calcule la moyenne arithmétique des valeurs des éléments	Statistiques
Median	Calcule la médiane des éléments matrice/vecteur	Statistiques
Min	Renvoie la valeur minimale dans une matrice/vecteur	Statistiques
Norm	Retourne la norme d'une matrice ou d'un vecteur	Fonctionnalités
Ones	Crée et renvoit une nouvelle matrice remplie de 1	Initialisation
Outer	Produit extérieur de deux matrices ou de deux vecteurs	Opérations
Percentile	Renvoie le centile spécifié des valeurs des éléments matriciels/vectoriels ou des éléments le long de l'axe spécifié	Statistiques
PInv	Calcule la pseudo-inverse d'une matrice par la méthode de Moore-Penrose	Solutions

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Catégorie
Power	Élevation d'une matrice carrée à une puissance entière	Opérations
Prod	Renvoie le produit des éléments matrice/vecteur, qui peut également être exécuté pour l'axe donné	Statistiques
Ptp	Renvoie la plage de valeurs d'une matrice/vecteur ou de l'axe de matrice donné	Statistiques
QR	Calcule la factorisation qr d'une matrice	Transformations
Quantile	Renvoie le quantile spécifié des valeurs des éléments matriciels/vectoriels ou des éléments le long de l'axe spécifié	Statistiques
Rank	Renvoie le rang de la matrice à l'aide de la méthode Gaussienne	Fonctionnalités
RegressionMetric	Calculer la métrique de régression en tant qu'erreur d'écart par rapport à la ligne de régression construite sur le tableau de données spécifié	Statistiques
Reshape	Modifie la forme d'une matrice sans modifier ses données	Manipulations
Resize	Renvoie une nouvelle matrice avec une forme et une taille modifiées	Manipulations
Row	Renvoie un vecteur ligne. Ecrit le vecteur dans la ligne spécifiée	Manipulations
Rows	Retourne le nombre de lignes d'une matrice	Fonctionnalités
Set	Définit la valeur de l'élément d'un vecteur en spécifiant son indice	Manipulations
Size	Retourne la taille d'un vecteur	Fonctionnalités
SLogDet	Calcule le signe et le logarithme du déterminant d'une matrice	Fonctionnalités
Solve	Résout une équation matricielle linéaire ou un système	Solutions

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Catégorie
	d'équations algébriques linéaires	
Sort	Trier en place	Manipulations
Spectrum	Calcule le spectre d'une matrice comme l'ensemble de ses valeurs propres du produit AT*A	Fonctionnalités
Split	Divise une matrice en plusieurs sous-matrices	Manipulations
Std	Renvoie l'écart type des valeurs des éléments matriciels/vectoriels ou des éléments le long de l'axe spécifié	Statistiques
Sum	Renvoie la somme des éléments de matrice/vecteur, qui peuvent également être exécutés pour l'axe (les axes) donné(s)	Statistiques
SVD	Décomposition en valeurs singulières	Transformations
SwapCols	Echange des colonnes dans une matrice	Manipulations
SwapRows	Échange des lignes dans une matrice	Manipulations
Trace	Retourne la somme des diagonales d'une matrice	Fonctionnalités
Transpose	Transpose (échanger les axes) et renvoit la matrice modifiée	Manipulations
Tri	Construit une matrice avec des 1 sur une diagonale spécifiée et en dessous, et des 0 ailleurs	Initialisation
TriL	Renvoie une copie d'une matrice avec des éléments au-dessus de la k-ième diagonale mis à 0. Matrice triangulaire inférieure	Manipulations
TriU	Renvoie une copie d'une matrice avec les éléments sous la k-ième diagonale mis à 0. Matrice triangulaire supérieure	Manipulations
Var	Calcule la variance des valeurs des éléments de matrice/vecteur	Statistiques

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Catégorie
Vsplit	Divise verticalement une matrice en plusieurs sous-matrices. Identique à Split avec axis=1	Manipulations
Zeros	Crée et renvoit une nouvelle matrice remplie de 0	Initialisation

© 2000-2024, MetaQuotes Ltd.

Types de matrice et de vecteur

La matrice et le vecteur sont des types de données spéciaux dans MQL5 qui permettent des opérations d'algèbre linéaire. Les types de données suivants existent :

- matrix — une matrice contenant des éléments de type double.
- matrixf — une matrice contenant des éléments de type float.
- matrixc — une matrice contenant des éléments de type complexe.
- vector — un vecteur contenant des éléments de type double.
- vectorf — un vecteur contenant des éléments de type float.
- vectorc — un vecteur contenant des éléments de type complexe.

Les fonctions templatées prennent en charge les notations telles que matrix<double>, matrix<float>, vector<double>, vector<float> au lieu des types correspondants.

Méthodes d'initialisation matricielle et vectorielle

Fonction	Action
Eye	Renvoie une matrice avec des 1 sur la diagonale et des 0 ailleurs
Identity	Crée une matrice d'identité de la taille spécifiée
Ones	Crée et renvoit une nouvelle matrice remplie de 1
Zeros	Crée et renvoit une nouvelle matrice remplie de 0
Full	Crée et retourne une nouvelle matrice remplie avec la valeur donnée
Tri	Construire une matrice avec des 1 sur la diagonale donnée et en dessous, et des 0 ailleurs
Init	Initialise une matrice ou un vecteur
Fill	Remplit une matrice ou un vecteur existant avec la valeur spécifiée

© 2000-2024, MetaQuotes Ltd.

Énumération pour les opérations sur les matrices et sur les vecteurs

Cette section décrit les énumérations utilisées dans les diverses méthodes matricielles et vectorielles.

ENUM_AVERAGE_MODE

Énumération des types de lissages.

ID	Description
AVERAGE_NONE	Pas de lissage. Les résultats sont fournis séparément pour chaque étiquette
AVERAGE_BINARY	Résultat de l'étiquette 1 pour une classification binaire
AVERAGE_MICRO	Résultat de la matrice d'erreur moyenne (matrice de confusion)
AVERAGE_MACRO	Résultat moyen à partir des résultats des matrices d'erreurs de chaque étiquette
AVERAGE_WEIGHTED	Résultat moyen pondéré

ENUM_VECTOR_NORM

Énumération des normes vectorielles pour vector::Norm.

ID	Description
VECTOR_NORM_INF	Norme Inf
VECTOR_NORM_MINUS_INF	-Norme Inf
VECTOR_NORM_P	Norme P

ENUM_MATRIX_NORM

Énumération des normes matricielles pour matrix::Norm et pour obtenir le conditionnement matriciel matrix::Cond.

ID	Description
MATRIX_NORM_FROBENIUS	Norme de Frobenius
MATRIX_NORM_SPECTRAL	Norme spectrale
MATRIX_NORM_NUCLEAR	Norme nucléaire
MATRIX_NORM_INF	Norme Inf

© 2000-2024, MetaQuotes Ltd.

ID	Description
MATRIX_NORM_P1	Norme P1
MATRIX_NORM_P2	Norme P2
MATRIX_NORM_MINUS_INF	-Norme Inf
MATRIX_NORM_MINUS_P1	-Norm P1
MATRIX_NORM_MINUS_P2	-Norm P2

ENUM_VECTOR_CONVOLVE

Énumération pour la convolution vector::Convolve et pour la corrélation croisée vector::Correlate.

ID	Description
VECTOR_CONVOLVE_FULL	Convolution complète
VECTOR_CONVOLVE_SAME	Convolution identique
VECTOR_CONVOLVE_VALID	Convolution valide

ENUM_REGRESSION_METRIC

Énumération des métriques de régression pour vector::RegressionMetric.

ID	Description
REGRESSION_MAE	Erreur Absolue Moyenne
REGRESSION_MSE	Erreur Quadratique Moyenne
REGRESSION_RMSE	Erreur Quadratique Moyenne
REGRESSION_R2	R au Carré
REGRESSION_MAPE	Erreur Absolue Moyenne en Pourcentage
REGRESSION_MSPE	Erreur Quadratique Moyenne en Pourcentage
REGRESSION_RMSLE	Erreur Logarithmique Quadratique Moyenne
REGRESSION_SMAPE	Erreur de Pourcentage Absolue Moyenne Symétrique
REGRESSION_MAXE	Erreur Absolue Maximale
REGRESSION_MEDE	Erreur Absolue Médiane
REGRESSION_MPD	Déviance Moyenne de Poisson
REGRESSION_MGD	Déviance Gamma Moyenne

© 2000-2024, MetaQuotes Ltd.

ID	Description
REGRESSION_EXPV	Écart Expliqué

ENUM_CLASSIFICATION_METRIC

Énumération de métriques pour les problèmes de classification.

ID	Description
CLASSIFICATION_ACCURACY	Qualité du modèle en termes de précision des prédictions pour toutes les classes
CLASSIFICATION_AVERAGE_PRECISION	Précision moyenne du modèle
CLASSIFICATION_BALANCED_ACCURACY	Précision de prédiction équilibrée
CLASSIFICATION_F1	Score F1. Moyenne harmonique entre la précision et le rappel du modèle
CLASSIFICATION_JACCARD	Score Jaccard
CLASSIFICATION_PRECISION	Précision du modèle dans la prédiction des vrais positifs pour la classe cible
CLASSIFICATION_RECALL	Complétude du modèle
CLASSIFICATION_ROC_AUC	Aire sous la courbe d'erreur
CLASSIFICATION_TOP_K_ACCURACY	Fréquence de l'étiquette correcte apparaissant en haut des k étiquettes prédites

ENUM_LOSS_FUNCTION

Énumération pour les calculs de la fonction de perte vector::Loss.

ID	Description
LOSS_MSE	Erreur quadratique moyenne
LOSS_MAE	Erreur absolue moyenne
LOSS_CCE	Crossentropie catégorielle
LOSS_BCE	Crossentropie binaire
LOSS_MAPE	Erreur absolue moyenne en pourcentage
LOSS_MSLE	Erreur logarithmique quadratique moyenne
LOSS_KLD	Divergence Kullback-Leibler
LOSS_COSINE	Cosinus similarité/proximité

© 2000-2024, MetaQuotes Ltd.

ID	Description
LOSS_POISSON	Poisson
LOSS_HINGE	Hinge
LOSS_SQ_HINGE	Hinge au carré
LOSS_CAT_HINGE	Hinge catégoriel
LOSS_LOG_COSH	Logarithme du cosinus hyperbolique
LOSS_HUBER	Huber

ENUM_ACTIVATION_FUNCTION

Énumération pour la fonction d'activation vector::Activation et pour la dérivée de la fonction d'activation vector::Derivative.

ID	Description
AF_ELU	Unité linéaire exponentielle
AF_EXP	Exponentiel
AF_GELU	Unité linéaire d'erreur gaussienne
AF_HARD_SIGMOID	Sigmoïde dur
AF_LINEAR	Linéaire
AF_LRELU	Unité linéaire rectifiée qui fuit
AF_RELU	Unité linéaire rectifiée
AF_SELU	Unité linéaire exponentielle mise à l'échelle
AF_SIGMOID	Sigmoïde
AF_SOFTMAX	Softmax
AF_SOFTPLUS	Softplus
AF_SOFTSIGN	Softsign
AF_SWISH	Swish
AF_TANH	La fonction tangente hyperbolique
AF_TRELU	Unité Linéaire Rectifiée Seuillée

ENUM_SORT_MODE

Énumération des types de tri pour la fonction Sort.

© 2000-2024, MetaQuotes Ltd.

ID	Description
SORT_ASCENDING	Tri ascendant
SORT_DESCENDING	Tri descendant

ENUM_MATRIX_AXIS

Enumération permettant de spécifier l'axe dans toutes les fonctions statistiques pour les matrices.

ID	Description
AXIS_NONE	L'axe n'est pas spécifié. Le calcul est effectué sur tous les éléments de la matrice, comme s'il s'agissait d'un vecteur (voir la méthode Flat).
AXIS_HORZ	Axe horizontal
AXIS_VERT	Axe vertical

© 2000-2024, MetaQuotes Ltd.

Initialisation

Il existe plusieurs façons de déclarer et d'initialiser des matrices et des vecteurs.

Fonction	Action
Assign	Copie une matrice, un vecteur ou un tableau avec une distribution automatique
CopyIndicatorBuffer	Retourne la quantité spécifiée de données du buffer spécifié de l'indicateur vers un vecteur
CopyRates	Récupère la série historique de la structure MqlRates pour le symbole et la période spécifiés, avec le nombre spécifié d'éléments dans une matrice ou un vecteur
CopyTicks	Récupère les ticks d'une structure MqlTick dans une matrice ou un vecteur
CopyTicksRange	Récupère les ticks d'une structure MqlTick dans une matrice ou un vecteur dans la plage de dates spécifiée
Eye	Renvoie une matrice avec des 1 sur la diagonale et des 0 ailleurs
Identity	Crée une matrice d'identité de la taille spécifiée
Ones	Crée et renvoit une nouvelle matrice remplie de 1
Zeros	Crée et renvoit une nouvelle matrice remplie de 0
Full	Crée et retourne une nouvelle matrice remplie avec la valeur donnée
Tri	Construire une matrice avec des 1 sur la diagonale donnée et en dessous, et des 0 ailleurs
Init	Initialise une matrice ou un vecteur
Fill	Remplit une matrice ou un vecteur existant avec la valeur spécifiée

Déclaration sans préciser de taille (pas d'allocation mémoire pour les données) :

```
matrix         matrix_a;   // matrice de types double
matrix<double> matrix_a1;  // une autre façon de déclarer une matrice de types doubl
matrixf        matrix_a2;  // matrice de types float
matrix<float>  matrix_a3;  // matrice de types float
vector         vector_a;   // matrice de types double
vector<double> vector_a1;
vectorf        vector_a2;  // matrice de types float
vector<float>  vector_a3;
```

Déclaration avec une taille spécifiée (avec allocation de mémoire pour les données, mais sans aucune initialisation) :

```
matrix         matrix_a(128,128);           // les paramètres peuvent être soit cons
```

© 2000-2024, MetaQuotes Ltd.

```
matrix<double> matrix_a1(InpRows,InpCols);  // soit variables
matrixf        matrix_a2(1,128);            // comme un vecteur horizontal
matrix<float>  matrix_a3(InpRows,1);        // comme un vecteur vertical
vector         vector_a(256);
vector<double> vector_a1(InpSize);
vectorf        vector_a2(SomeFunc());       // la fonction SomeFunc retourne une val
vector<float>  vector_a3(InpSize+16);       // l'expression peut être utilisée comme
```

Déclaration avec initialisation (les tailles de matrice et de vecteur sont déterminées par la séquence d'initialisation) :

```
matrix         matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
matrix<double> matrix_a1=matrix_a;                      // les matrices doivent être
matrixf        matrix_a2={{1,0,0},{0,1,0},{0,0,1}};
matrix<float>  matrix_a3={{1,2},{3,4}};
vector         vector_a={-5,-4,-3,-2,-1,0,1,2,3,4,5};
vector<double> vector_a1={1,5,2.4,3.3};
vectorf        vector_a2={0,1,2,3};
vector<float>  vector_a3=vector_a2;                     // les vecteurs doivent être
```

Déclaration avec initialisation :

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
...

  matrix  matrix_a(size_m,size_k,MatrixArange,-M_PI,0.1); // une matrice non initialis
  matrixf matrix_a1(10,20,MatrixArange);                  // après avoir créé la matri
  vector  vector_a(size,VectorArange,-10.0);              // après avoir le vecteurl l
  vectorf vector_a1(128,VectorArange);
```

© 2000-2024, MetaQuotes Ltd.

Veuillez noter que les dimensions de la matrice ou du vecteur peuvent être changés, car la mémoire allouée pour les données est toujours dynamique.

Méthodes statiques

Méthodes statiques pour créer des matrices et des vecteurs de la taille spécifiée, initialisés d'une certaine façon :

```
matrix          matrix_a =matrix::Eye(4,5,1);
matrix<double> matrix_a1=matrix::Full(3,4,M_PI);
matrixf         matrix_a2=matrixf::Identity(5,5);
matrixf<float> matrix_a3=matrixf::Ones(5,5);
matrix          matrix_a4=matrix::Tri(4,5,-1);
vector          vector_a =vector::Ones(256);
vectorf         vector_a1=vector<float>::Zeros(16);
vector<float>   vector_a2=vectorf::Full(128,float_value);
```

Méthodes pour initialiser des matrices et des vecteurs déjà créés :

```
matrix  matrix_a;
matrix_a.Init(size_m,size_k,MatrixArange,-M_PI,0.1);
matrixf matrix_a1(3,4);
matrix_a1.Init(10,20,MatrixArange);
vector  vector_a;
vector_a.Init(128,VectorArange);
vectorf vector_a1(10);
vector_a1.Init(vector_size,VectorArange,start_value,step);

matrix_a.Fill(double_value);
vector_a1.Fill(FLT_MIN);
matrix_a1.Identity();
```

© 2000-2024, MetaQuotes Ltd.

Assign

Copie une matrice, un vecteur ou un tableau avec une distribution automatique.

```
bool matrix::Assign(
  const matrix<T>  &mat     // matrice copiée
   );
bool matrix::Assign(
  const void       &array[] // tableau copié
   );
bool vector::Assign(
  const vector<T>  &vec     // vecteur copié
   );
bool vector::Assign(
  const void       &array[] // tableau copié
   );
```

Paramètres

m, v ou tableau

[in] La matrice, le vecteur ou le tableau à partir duquel les valeurs sont copiées.

Valeur de Retour

Renvoie true en cas de succès, sinon false.

Note

Contrairement à Copy, la méthode Assign permet également de copier des tableaux. Dans ce cas, la conversion automatique a lieu, et la matrice ou le vecteur résultant s'ajuste à la taille du tableau copié.

Exemple :

```
//--- copie les matrices
  matrix a= {{2, 2}, {3, 3}, {4, 4}};
  matrix b=a+2;
  matrix c;
  Print("matrix a \n", a);
  Print("matrix b \n", b);
  c.Assign(b);
  Print("matrix c \n", a);

//--- copie le tableau dans la matrice
  matrix double_matrix=matrix::Full(2,10,3.14);
  Print("double_matrix avant Assign() \n", double_matrix);
  int int_arr[5][5]= {{1, 2}, {3, 4}, {5, 6}};
  Print("int_arr: ");
  ArrayPrint(int_arr);
```

© 2000-2024, MetaQuotes Ltd.

```
double_matrix.Assign(int_arr);
Print("double_matrix après Assign(int_arr) \n", double_matrix);
/*
 matrix a
 [[2,2]
  [3,3]
  [4,4]]
 matrix b
 [[4,4]
  [5,5]
  [6,6]]
 matrix c
 [[2,2]
  [3,3]
  [4,4]]

 double_matrix avant Assign()
 [[3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]
  [3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]]

 int_arr:
     [,0][,1][,2][,3][,4]
 [0,]   1   2   0   0   0
 [1,]   3   4   0   0   0
 [2,]   5   6   0   0   0
 [3,]   0   0   0   0   0
 [4,]   0   0   0   0   0

 double_matrix après Assign(int_arr)
 [[1,2,0,0,0]
  [3,4,0,0,0]
  [5,6,0,0,0]
  [0,0,0,0,0]
  [0,0,0,0,0]]

*/
```

Voir aussi

Copy

© 2000-2024, MetaQuotes Ltd.

CopyIndicatorBuffer

Retourne la quantité spécifiée de données du buffer spécifié de l'indicateur vers un vecteur.

Les données seront copiées dans le vecteur avec le plus vieil élément situé au début de la mémoire physique allouée au vecteur. La fonction existe sous 3 formes.

Accès par la position initiale et le nombre d'éléments requis

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle de l'indicateur
   ulong     buffer_index,         // numéro de buffer de l'indicateur
   ulong     start_pos,            // indice de départ de la copie
   ulong     count                 // nombre d'éléments à copier
   );
```

Accès par date de début et nombre d'éléments requis

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle de l'indicateur
   ulong     buffer_index,         // numéro de buffer de l'indicateur
   datetime  start_time,           // à partir de quelle date copier
   ulong     count                 // nombre d'éléments à copier
   );
```

Accès aux dates initiale et finale de l'intervalle de temps requis

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle de l'indicateur
   ulong     buffer_index,         // numéro de buffer de l'indicateur
   datetime  start_time,           // à partir de quelle date copier
   datetime  stop_time             // jusqu'à quelle date copier
   );
```

Paramètres

indicator_handle

[in] Le handle d'indicateur obtenu par la fonction de l'indicateur.

buffer_index

[in] Le numéro du buffer de l'indicateur.

start_pos

[in] Le numéro du premier index d'élément copié.

count

[in] Le nombre d'éléments à copier.

start_time

[in] Heure de la barre correspondant au premier élément.

stop_time

[in] Heure de la barre correspondant au dernier élément.

© 2000-2024, MetaQuotes Ltd.

Valeur de Retour

La fonction renvoie 'true' en cas de succès ou 'false' si une erreur se produit.

Note

Les éléments des données (le buffer de l'indicateur d'index buffer_index) copiées sont décomptés du présent au passé. La position de départ égale à 0 signifie donc la barre actuelle (la valeur de l'indicateur pour la barre actuelle).

Lors de la copie d'une quantité inconnue de données, vous devez déclarer un vecteur sans spécifier de taille (sans allouer de mémoire pour les données), car la fonction CopyBuffer() essaie d'adapter la taille du vecteur destination à la taille des données copiées.

Lorsqu'une copie partielle des valeurs de l'indicateur est requise, vous devriez utiliser un vecteur intermédiaire dans lequel la quantité nécessaire de données est copiée. A partir de ce vecteur intermédiaire, vous pouvez copier le nombre requis de valeurs membre par membre, aux emplacements requis du vecteur destination.

Si vous copiez une quantité prédéterminée de données, il est recommandé de pré-déclarer un vecteur et de spécifier sa taille pour éviter une réallocation de mémoire inutile.

Lors de la demande de données à partir d'un indicateur, la fonction renvoie immédiatement false si les séries temporelles demandées n'ont pas encore été construites ou doivent être téléchargées depuis le serveur, pendant qu'elle initie le chargement/la construction.

Lors de la demande de données à partir d'un EA ou d'un script, le téléchargement depuis le serveur est lancé si le terminal ne dispose pas des données appropriées localement, ou la construction des séries temporelles nécessaires commence si les données peuvent être construites à partir de l'historique local mais que les délais requis ne sont pas encore prêts. La fonction retourne la quantité qui sera prête à l'expiration du délai.

Voir également

CopyBuffer

© 2000-2024, MetaQuotes Ltd.

CopyRates

Récupère la série historique de la structure MqlRates pour le symbole et la période spécifiés, avec le nombre spécifié d'éléments dans une matrice ou un vecteur. Les éléments sont décomptés du présent au passé, ce qui signifie que la position de départ égale à 0 signifie la barre actuelle.

Les données sont copiées de manière à ce que l'élément le plus ancien soit placé au début de la matrice/du vecteur. La fonction existe sous 3 formes.

Accès par la position initiale et le nombre d'éléments requis

```
bool  CopyRates(
   string           symbol,       // nom du symbole
   ENUM_TIMEFRAMES  period,       // période
   ulong            rates_mask,   // combinaison de flags pour spécifier les séries de
   ulong            start,        // indice de la barre initiale d'où la copie commenc
   ulong            count         // nombre d'éléments à copier
   );
```

Accès par la date de début et le nombre d'éléments requis

```
bool  CopyRates(
   string           symbol,       // nom du symbole
   ENUM_TIMEFRAMES  period,       // période
   ulong            rates_mask,   // combinaison de flags pour spécifier les séries de
   datetime         from,         / date de début
   ulong            count         // nombre d'éléments à copier
   );
```

Accès par les dates de début et de fin de l'intervalle de temps désiré

```
bool  CopyRates(
   string           symbol,       // nom du symbole
   ENUM_TIMEFRAMES  period,       // période
   ulong            rates_mask,   // combinaison de flags pour spécifier les séries de
   datetime         from,         / date de début
   datetime         to            // date de fin
   );
```

Paramètres

symbol

[in] Symbole.

period

[in] Période.

rates_mask

[in] La combinaison d'indicateurs d'énumération ENUM_COPY_RATES spécifiant le type de série demandée. Lors de la copie vers un vecteur, une seule valeur de l'énumération ENUM_COPY_RATES peut être spécifiée, sinon une erreur se produit.

start

© 2000-2024, MetaQuotes Ltd.

[in] Indice du premier élément à copier.

count

[in] Nombre d'éléments à copier.

from

[in] Barre du temps correspondant au premier élément.

to

[in] Barre du temps correspondant au dernier élément.

Valeur de Retour

Renvoie true en cas de succès, sinon false en cas d'erreur.

Note

Si l'intervalle des données demandées est complètement en dehors des données disponibles sur le serveur, alors la fonction renvoie false. Si les données en dehors de TERMINAL_MAXBARS (nombre maximum de barres sur le graphique) sont demandées, la fonction renvoie également false.

Lors de la demande de données à partir d'un EA ou d'un script, le téléchargement depuis le serveur est lancé si le terminal ne dispose pas des données appropriées localement, ou la construction des séries temporelles nécessaires commence si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction renvoie la quantité qui sera prête à l'expiration du délai d'attente. Mais le téléchargement de l'historique se poursuit et la fonction renverra plus de données au prochain appel.

Lors de la demande de données par la date de début et par le nombre d'éléments requis, seules les données dont la date est inférieure (avant) ou égale à celle spécifiée sont renvoyées. L'intervalle est défini et considéré jusqu'à une seconde. En d'autres termes, la date d'ouverture de toute barre pour laquelle la valeur est renvoyée (volume, écart, ouverture, haut, bas, clôture ou heure) est toujours égale ou inférieure à celle spécifiée.

Lors de la demande de données dans une plage de dates donnée, seules les données comprises dans l'intervalle demandé sont renvoyées. L'intervalle est défini et considéré jusqu'à une seconde. En d'autres termes, l'heure d'ouverture de toute barre pour laquelle la valeur est renvoyée (volume, spread, valeur du buffer de l'indicateur, ouverture, haut, bas, clôture ou temps) est toujours située dans l'intervalle demandé.

Par exemple, si le jour actuel de la semaine est le samedi, la fonction renvoie 0 lorsqu'elle tente de copier des données sur la période hebdomadaire en définissant *start_time=Last_Tuesday* et *stop_time=Last_Friday* car l'heure d'ouverture sur la période hebdomadaire tombe toujours le dimanche, mais pas une seule barre hebdomadaire ne se situe dans la plage spécifiée.

Si vous avez besoin d'obtenir la valeur correspondant à la barre incomplète actuelle, vous pouvez utiliser le premier formulaire d'appel indiquant *start_pos*=0 et *count*=1.

ENUM_COPY_RATES

L'énumération ENUM_COPY_RATES contient les indicateurs permettant de spécifier le type de données à transmettre à la matrice ou au tableau. Le paramètre de combinaison de flags permet d'obtenir plusieurs séries de l'historique en une seule requête. L'ordre des lignes dans la matrice correspondra à

© 2000-2024, MetaQuotes Ltd.

l'ordre des valeurs dans l'énumération ENUM_COPY_RATES. En d'autres termes, la ligne avec des données Plus Haut sera toujours plus élevée dans la matrice que la ligne avec les données Plus Bas.

ID	Valeur	Description
COPY_RATES_OPEN	1	Série des prix d'ouverture
COPY_RATES_HIGH	2	Série des prix Plus Haut
COPY_RATES_LOW	4	Série des prix Plus Bas
COPY_RATES_CLOSE	8	Série des prix de clôture
COPY_RATES_TIME	16	Séries des dates et heures (heure d'ouverture des barres) Récupérer les dates et heures dans un vecteur ou une matrice d'éléments de type float (vectord et matrixf) entraîne des pertes d'environ 100 secondes puisque la précision du type float est limitée et les nombres entiers supérieurs à 1<<24 ne peuvent pas être représentés avec précision en float.
COPY_RATES_VOLUME_TICK	32	Volumes des Ticks
COPY_RATES_VOLUME_REAL	64	Volumes des Trades
COPY_RATES_SPREAD	128	Spreads
Combinaison		
COPY_RATES_OHLC	15	Séries OHLC (Ouverture, Plus Haut, Plus Bas et Clôture)
COPY_RATES_OHLCT	31	Séries OHLC (Ouverture, Plus Haut, Plus Bas et Clôture) + Temps

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- obtient les cotations dans la matrice
  matrix matrix_rates;
  if(matrix_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLCT, 1, 10))
    Print("matrix rates: \n", matrix_rates);
  else
    Print("Echec de matrix_rates.CopyRates. Erreur ", GetLastError());
//--- vérification
  MqlRates mql_rates[];
  if(CopyRates(Symbol(), PERIOD_CURRENT, 1, 10, mql_rates)>0)
```

© 2000-2024, MetaQuotes Ltd.

```
    {
     Print("Tableau mql_rates :");
     ArrayPrint(mql_rates);
    }
  else
    Print("CopyRates(Symbol(), PERIOD_CURRENT,1, 10, mql_rates). Erreur ", GetLastErro
//--- obtient les cotations dans le vecteur = appel invalide
  vector vector_rates;
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLC, 1, 15))
    Print("vector_rates COPY_RATES_OHLC: \n", vector_rates);
  else
    Print("vector_rates.CopyRates COPY_RATES_OHLC failed. Erreur ", GetLastError());
//--- obtient les prix de clôture dans le vecteur
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_CLOSE, 1, 15))
    Print("vector_rates COPY_RATES_CLOSE: \n", vector_rates);
  else
    Print("Echec de vector_rates.CopyRates. Erreur ", GetLastError());
 };
/*
   matrix rates:
   [[0.99686,0.99638,0.99588,0.99441,0.99464,0.99594,0.99698,0.99758,0.99581,0.995280
    [0.99708,0.99643,0.99591,0.995500000000001,0.99652,0.99795,0.99865,0.99764,0.996
    [0.9961100000000001,0.99491,0.99426,0.99441,0.99448,0.99494,0.9964499999999999,0.9
    [0.99641,0.99588,0.99441,0.99464,0.99594,0.99697,0.99758,0.99581,0.995280000000000
    [1662436800,1662440400,1662444000,1662447600,1662451200,1662454800,1662458400,1662
   mql_rates array:
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2022.09.06 04:00:00 0.99686 0.99708 0.99611 0.99641          4463        0
   [1] 2022.09.06 05:00:00 0.99638 0.99643 0.99491 0.99588          4519        0
   [2] 2022.09.06 06:00:00 0.99588 0.99591 0.99426 0.99441          3060        0
   [3] 2022.09.06 07:00:00 0.99441 0.99550 0.99441 0.99464          3867        0
   [4] 2022.09.06 08:00:00 0.99464 0.99652 0.99448 0.99594          5280        0
   [5] 2022.09.06 09:00:00 0.99594 0.99795 0.99494 0.99697          7227        0
   [6] 2022.09.06 10:00:00 0.99698 0.99865 0.99645 0.99758         10130        0
   [7] 2022.09.06 11:00:00 0.99758 0.99764 0.99472 0.99581          7012        0
   [8] 2022.09.06 12:00:00 0.99581 0.99604 0.99360 0.99528          6166        0
   [9] 2022.09.06 13:00:00 0.99528 0.99570 0.99220 0.99259          6950        0
   vector_rates.CopyRates COPY_RATES_OHLC failed. Erreur 4003
   vector_rates COPY_RATES_CLOSE:
   [0.9931,0.99293,0.99417,0.99504,0.9968399999999999,0.99641,0.99588,0.99441,0.99464,
*/
```

Voir aussi

Accéder aux Timeseries et aux Indicateurs, CopyRates

© 2000-2024, MetaQuotes Ltd.

CopyTicks

Retourne les ticks d'une structure MqlTick dans une matrice ou dans un vecteur. Les éléments sont comptés du passé vers le présent, ce qui signifie que l'élément à l'indice 0 est le plus ancien. Pour analyser un tick, vérifiez le champ *flags* qui montre exactement ce qui a changé dans le tick.

```
bool  matrix::CopyTicks(
   string          symbol,                // nom du symbole
   ulong           ticks_mask,            // masque indiquant le type de ticks désiré
   uint            flags=COPY_TICKS_ALL,  // drapeau indiquant le type de ticks à rec
   ulong           from_msc=0,            // heure à partir de laquelle les ticks son
   ulong           count=0                // nombre de ticks à obtenir
   );
```

Méthode retournant un vecteur

```
bool  vector::CopyTicks(
   string          symbol,                // nom du symbole
   ulong           ticks_mask,            // masque indiquant le type de ticks désiré
   uint            flags=COPY_TICKS_ALL,  // drapeau indiquant le type de ticks à rec
   ulong           from_msc=0,            // heure à partir de laquelle les ticks son
   ulong           count=0                // nombre de ticks à obtenir
   );
```

Paramètres

symbol

[in] Symbole.

ticks_mask

[in] Une combinaison de drapeaux de l'énumération ENUM_COPY_TICKS indiquant le contenu des données demandées. Lors de la copie vers un vecteur, vous ne pouvez spécifier qu'une seule valeur de l'énumération ENUM_COPY_TICKS, sinon une erreur sera retournée.

flags

[in] Un drapeau définissant le type des ticks demandés. COPY_TICKS_INFO signifie tous les ticks concernés par un changement des prix Bid et/ou Ask, COPY_TICKS_TRADE — signifie tous les ticks avec un changement du prix Last (dernier prix) ou du Volume, COPY_TICKS_ALL — signifie tous les ticks. Pour toute demande, les valeurs du tick précédent sont ajoutées aux champs restants de la structure MqlTick.

from_msc

[in] Heure à partir de laquelle les ticks sont demandés. L'heure est spécifiée en millisecondes depuis le 01/01/1970. Si from_msc=0, le dernier nombre de ticks égal à 'count' est renvoyé.

count

[in] Le nombre de ticks demandés. Si les paramètres 'from_msc' et 'count' ne sont pas spécifiés, tous les ticks disponibles, mais pas plus de 2000, seront écrits.

© 2000-2024, MetaQuotes Ltd.

Valeur de Retour

Renvoie true en cas de succès ou false si une erreur se produit.

Note

Le premier appel à CopyTicks() lance la synchronisation de la base de données de ticks du symbole concerné stockée sur le disque dur. Si la base de données locale ne peut pas fournir tous les ticks demandés, les ticks manquants sont automatiquement téléchargés depuis le serveur de trading. Les ticks depuis ***from_msc*** spécifié dans CopyTicks() jusqu'au moment présent seront synchronisés. Après cela, tous les ticks arrivant pour ce symbole seront ajoutés à la base de données de ticks, le maintenant ainsi dans un état synchronisé.

Si les paramètres *from_msc* et *count* ne sont pas spécifiés, tous les ticks disponibles, mais pas plus de 2000, seront écrits dans la matrice ou dans le vecteur.

Pour les indicateurs, la méthode CopyTicks() retourne le résultat immédiatement : lorsqu'elle est appelée depuis un indicateur, la méthode CopyTick() retourne immédiatement tous les ticks disponibles du symbole et lance la synchronisation de la base de données des ticks s'il n'y a pas assez de données. Tous les indicateurs d'un même symbole opérent dans un seul thread commun, l'indicateur ne peut donc pas attendre la fin de la synchronisation. Après la synchronisation, CopyTicks() renverra tous les ticks demandés lors du prochain appel. Dans les indicateurs, la fonction OnCalculate() est appelée après l'arrivée de chaque tick.

Dans les Expert Advisors et les scripts, CopyTicks() peut attendre le résultat pendant 45 secondes : contrairement aux indicateurs, chaque Expert Advisor ou script fonctionne dans un thread séparé et peut donc attendre jusqu'à 45 secondes pour que la synchronisation se termine. Si la quantité requise de ticks ne parvient pas à être synchronisée pendant ce temps, CopyTicks() renverra les ticks disponibles au bout du délai imparti (timeout) et poursuivra la synchronisation. OnTick() dans Expert Advisors n'est pas un gestionnaire de chaque tick, il informe uniquement l'Expert Advisor des changements sur le marché. Il peut s'agir d'un lot de changements : le terminal peut recevoir simultanément plusieurs ticks, tandis que OnTick() ne sera appelé qu'une seule fois, pour informer l'Expert Advisor du dernier état du marché.

Taux de retour des données : le terminal stocke les 4 096 derniers ticks pour chaque instrument dans le cache à accès rapide (65 536 ticks pour les symboles avec la Profondeur du Marché en cours d'exécution). Les demandes concernant ces données sont celles exécutées le plus rapidement. Si les ticks demandés pour la session de trading en cours dépassent le cache, CopyTicks() appelle les ticks stockés dans la mémoire du terminal. Ces demandes nécessite plus de temps pour être terminées. Les demandes les plus lentes sont celles demandant des ticks des jours passés, puisque les données sont lues depuis le disque dur dans ce cas.

ENUM_COPY_TICKS

L'énumération ENUM_COPY_TICKS contient les indicateurs permettant de spécifier le type de données à transmettre à la matrice ou au tableau. Le paramètre de combinaison de flags permet d'obtenir plusieurs séries de l'historique en une seule requête. L'ordre des lignes dans la matrice correspondra à l'ordre des valeurs dans l'énumération ENUM_COPY_TICKS. En d'autres termes, la ligne avec des données Plus Haut sera toujours plus élevée dans la matrice que la ligne avec les données Plus Bas.

ID	Valeur	Description
COPY_TICKS_TIME_MS	1	L'heure du tick en millisecondes

© 2000-2024, MetaQuotes Ltd.

ID	Valeur	Description
COPY_TICKS_BID	2	Price Bid
COPY_TICKS_ASK	4	Prix Ask
COPY_TICKS_LAST	8	Dernier prix (prix de la dernière transaction)
COPY_TICKS_VOLUME	16	Volume du dernier prix
COPY_TICKS_FLAGS	32	Drapeaux des Ticks

Analysez les indicateurs des ticks pour savoir quelles données ont changé :

- TICK_FLAG_BID — le prix de l'Offre (Bid) du tick a changé
- TICK_FLAG_ASK — le prix de la Demande (Ask) du tick a changé
- TICK_FLAG_LAST — le prix de la dernière transaction du tick a changé
- TICK_FLAG_VOLUME — le volume du tick a changé
- TICK_FLAG_BUY — le tick est le résultat d'une opération d'achat
- TICK_FLAG_SELL — le tick est le résultat d'une opération de vente

Voir aussi

Accès aux séries chronologiques et aux indicateurs, CopyTicks

© 2000-2024, MetaQuotes Ltd.

CopyTicksRange

Récupère les ticks d'une structure MqlTick et les copie dans une matrice ou dans un vecteur dans la plage de dates spécifiée. Les éléments sont comptés du passé au présent, ce qui signifie que le tick d'index 0 est le plus ancien. Pour analyser un tick, vérifiez le champ *flags* qui montre exactement ce qui a changé dans le tick.

```
bool  matrix::CopyTicksRange(
   string          symbol,                // nom du symbole
   ulong           ticks_mask,            // masque indiquant le type de ticks désiré
   uint            flags=COPY_TICKS_ALL,  // drapeau indiquant le type de ticks à rec
   ulong           from_msc=0,            // heure à partir de laquelle les ticks son
   ulong           to_msc=0               // heure jusqu'à laquelle les ticks sont de
   );
```

Méthode retournant un vecteur

```
bool  vector::CopyTicksRange(
   string          symbol,                // nom du symbole
   ulong           ticks_mask,            // masque indiquant le type de ticks désiré
   uint            flags=COPY_TICKS_ALL,  // drapeau indiquant le type de ticks à rec
   ulong           from_msc=0,            // heure à partir de laquelle les ticks son
   ulong           to_msc=0               // heure jusqu'à laquelle les ticks sont de
   );
```

Paramètres

symbol

[in] Symbole.

ticks_mask

[in] Une combinaison de drapeaux de l'énumération ENUM_COPY_TICKS indiquant le contenu des données demandées. Lors de la copie vers un vecteur, vous ne pouvez spécifier qu'une seule valeur de l'énumération ENUM_COPY_TICKS, sinon une erreur sera retournée.

flags

[in] Un drapeau définissant le type des ticks demandés. COPY_TICKS_INFO signifie tous les ticks concernés par un changement des prix Bid et/ou Ask, COPY_TICKS_TRADE — signifie tous les ticks avec un changement du prix Last (dernier prix) ou du Volume, COPY_TICKS_ALL — signifie tous les ticks. Pour toute demande, les valeurs du tick précédent sont ajoutées aux champs restants de la structure MqlTick.

from_msc

[in] Heure à partir de laquelle les ticks sont demandés. L'heure est spécifiée en millisecondes depuis le 01/01/1970. Si le paramètre 'from_msc' n'est pas spécifié, les ticks depuis le début de l'historique sont renvoyés. Les ticks avec une heure >= from_msc seront retournés.

to_msc

[in] Heure jusqu'à laquelle les ticks sont demandés. L'heure est spécifiée en millisecondes depuis le 01/01/1970. Les ticks avec une heure <= to_msc sont retournés. Si le paramètre to_msc n'est pas spécifié, tous les ticks jusqu'à la fin de l'historique sont renvoyés.

© 2000-2024, MetaQuotes Ltd.

Valeur de Retour

Retourne true en cas de succès, ou false si une erreur s'est produite. GetLastError() peut retourner les erreurs suivantes :

- ERR_HISTORY_TIMEOUT — le délai d'attente pour la synchronisation des ticks a expiré, la fonction a renvoyé tout ce qu'elle avait.
- ERR_HISTORY_SMALL_BUFFER — le tampon statique est trop petit. Seule la quantité que le tableau peut stocker a été renvoyée.
- ERR_NOT_ENOUGH_MEMORY — mémoire insuffisante pour recevoir les données historiques de la plage spécifiée dans un tableau de ticks dynamique. Il n'avait pas asez de mémoire pouvant être allouée pour le tableau de ticks.

Analysez les indicateurs des ticks pour savoir quelles données ont changé :

- TICK_FLAG_BID — le prix de l'Offre (Bid) du tick a changé
- TICK_FLAG_ASK — le prix de la Demande (Ask) du tick a changé
- TICK_FLAG_LAST — le prix de la dernière transaction du tick a changé
- TICK_FLAG_VOLUME — le volume du tick a changé
- TICK_FLAG_BUY — le tick est le résultat d'une opération d'achat
- TICK_FLAG_SELL — le tick est le résultat d'une opération de vente

Note

La méthode CopyTicksRange() est utilisée pour demander les ticks d'un intervalle de temps spécifique. Par exemple, les ticks d'un jours spécifique de l'historique. CopyTicks() permet de spécifier uniquement la date de début, par exemple, pour recevoir tous les ticks depuis le début du mois jusqu'à maintenant.

Voir aussi

Accès aux séries chronologiques et aux indicateurs, CopyTicksRange

© 2000-2024, MetaQuotes Ltd.

Eye

Fonction statique. Construit une matrice ayant une taille spécifiée avec des 1 sur la diagonale principale et des 0 ailleurs. Retourne une matrice avec des 1 sur la diagonale et des 0 ailleurs.

```
static matrix matrix::Eye(
  const ulong  rows,        // nombre de lignes
  const ulong  cols,        // nombre de colonnes
  const int    ndiag=0      // indice de la diagonale
   );
```

Paramètres

rows

[in] Nombre de lignes dans le résultat.

cols

[in] Nombre de colonnes dans le résultat.

ndiag=0

[dans] Indice de la diagonale : 0 (valeur par défaut) fait référence à la diagonale principale, une valeur positive fait référence à une diagonale supérieure et une valeur négative à une diagonale inférieure.

Valeur de Retour

Une matrice où tous les éléments sont égaux à 0, à l'exception de la k-ième diagonale, dont les valeurs sont égales à 1.

Exemple en MQL5 :

```
  matrix eye=matrix::Eye(3, 3);
  Print("eye = \n", eye);

  eye=matrix::Eye(4, 4,1);
  Print("eye = \n", eye);
  /*
   eye =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
   eye =
   [[0,1,0,0]
    [0,0,1,0]
    [0,0,0,1]
    [0,0,0,0]]
  */
```

Exemple en Python :

```
np.eye(3, dtype=int)
```

© 2000-2024, MetaQuotes Ltd.

```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])

 np.eye(4, k=1)
array([[0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 0., 0.]])
```

© 2000-2024, MetaQuotes Ltd.

Identity

C'est une fonction statique qui crée une matrice identité de la taille spécifiée (pas nécessairement carrée). Une matrice d'identité contient des 1 sur la diagonale principale et des 0 ailleurs. La diagonale principale est constituée des éléments de la matrice ayant des indices de ligne et de colonne égaux, tels que [0,0], [1,1], [2,2] etc. Crée une nouvelle matrice identité.

Il existe également la méthode Identity qui transforme une matrice existante en matrice identité.

```
static matrix matrix::Identity(
  const ulong  rows,         // nombre de lignes
  const ulong  cols,         // nombre de colonnes
   );

void matrix::Identity();
```

Paramètres

rows

[in] Nombre de lignes (et de colonnes) dans la matrice n x n.

Valeur de Retour

Retourne la matrice identité. La matrice identité est une matrice carrée avec des 1 sur la diagonale principale.

Exemple en MQL5 :

```
  matrix identity=matrix::Identity(3,3);
  Print("identity = \n", identity);
/*
   identity =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
*/
  matrix identity2(3,5);
  identity2.Identity();
  Print("identity2 = \n", identity2);
/*
   identity2 =
   [[1,0,0,0,0]
    [0,1,0,0,0]
    [0,0,1,0,0]]
*/
```

Exemple en Python :

```
np.identity(3)
```

© 2000-2024, MetaQuotes Ltd.

```
array([[1.,  0.,  0.],
       [0.,  1.,  0.],
       [0.,  0.,  1.]])
```

© 2000-2024, MetaQuotes Ltd.

Ones

Il s'agit d'une fonction statique qui crée et renvoie une nouvelle matrice remplie de 1.

```
static matrix matrix::Ones(
  const ulong  rows,      // nombre de lignes
  const ulong  cols       // nombre de colonnes
   );

static vector vector::Ones(
  const ulong  size,
   );
```

Paramètres

rows

[in] Nombre de lignes.

cols

[in] Nombre de colonnes.

Valeur de Retour

Une nouvelle matrice du nombre de lignes et de colonnes spécifiés, remplie de 1.

Exemple en MQL5 :

```
  matrix ones=matrix::Ones(4, 4);
  Print("ones = \n", ones);
/*
ones =
   [[1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]]
*/
```

Exemple en Python :

```
np.ones((4, 1))
array([[1.],
       [1.]])
```

© 2000-2024, MetaQuotes Ltd.

Zeros

Il s'agit d'une fonction statique qui crée et renvoie une nouvelle matrice remplie de 0.

```
static matrix matrix::Zeros(
  const ulong  rows,      // nombre de lignes
  const ulong  cols       // nombre de colonnes
   );

static vector vector::Zeros(
  const ulong  size,
   );
```

Paramètres

rows

[in] Nombre de lignes.

cols

[in] Nombre de colonnes.

Valeur de Retour

Une nouvelle matrice de lignes et de colonnes données, remplie de 0.

Exemple en MQL5 :

```
  matrix zeros=matrix::Zeros(3, 4);
  Print("zeros = \n", zeros);
/*
zeros =
   [[0,0,0,0]
    [0,0,0,0]
    [0,0,0,0]]
*/
```

Exemple en Python :

```
np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

© 2000-2024, MetaQuotes Ltd.

Full

La fonction statique crée et renvoie une nouvelle matrice remplie d'une valeur donnée.

```
static matrix matrix::Full(
  const ulong   rows,       // nombre de lignes
  const ulong   cols,       // nombre de colonnes
  const double  value       // valeur de remplissage
   );

static vector vector::Full(
  const ulong   size,
  const double  value       // valeur de remplissage
   );
```

Paramètres

rows

[in] Nombre de lignes.

cols

[in] Nombre de colonnes.

value

[in] Valeur avec laquelle les éléments de la matrice seront remplis.

Valeur de Retour

Retourne une nouvelle matrice du nombre de lignes et de colonnes spécifiés, remplie avec la valeur spécifiée.

Exemple en MQL5 :

```
  matrix full=matrix::Full(3,4,10);
  Print("full = \n", full);
/*
full =
   [[10,10,10,10]
    [10,10,10,10]
    [10,10,10,10]]
*/
```

Exemple

```
np.full((2, 2), 10)
array([[10, 10],
       [10, 10]])
```

© 2000-2024, MetaQuotes Ltd.

Tri

Il s'agit d'une fonction statique. Construit une matrice avec des 1 sur la diagonale donnée et en dessous, et des 0 ailleurs.

```
static matrix matrix::Tri(
  const ulong rows,         // nombre de lignes
  const ulong cols,         // nombre de colonnes
  const int   ndiag=0       // nombre de diagonales
   );
```

Paramètres

rows

[in] Nombre de lignes dans la matrice.

cols

[in] Nombre de colonnes dans la matrice.

ndiag=0

[in] La sous-diagonale à laquelle et en dessous de laquelle la matrice est remplie. k = 0 est la diagonale principale, k < 0 signifie une diagonale en-dessous et k > 0 une diagonale au-dessus. La valeur par défaut est 0.

Valeur de Retour

Matrice avec son triangle inférieur rempli de 1 et 0 ailleurs.

Exemple en MQL5 :

```
   matrix matrix_a=matrix::Tri(3,4,1);
   Print("Tri(3,4,1)\n",matrix_a);
   matrix_a=matrix::Tri(4,3,-1);
   Print("Tri(4,3,-1)\n",matrix_a);

/*
   Tri(3,4,1)
   [[1,1,0,0]
    [1,1,1,0]
    [1,1,1,1]]
   Tri(4,3,-1)
   [[0,0,0]
    [1,0,0]
    [1,1,0]
    [1,1,1]]
*/
```

Exemple

```
np.tri(3, 5, 2, dtype=int)
array([[1, 1, 1, 0, 0],
       [1, 1, 1, 1, 0],
```

© 2000-2024, MetaQuotes Ltd.

```
        [1, 1, 1, 1, 1]])
```

© 2000-2024, MetaQuotes Ltd.

Init

Initialise une matrice ou un vecteur.

```
void matrix::Init(
  const ulong  rows,            // nombre de lignes
  const ulong  cols,            // nombre de colonnes
  func_name    init_func=NULL,  // fonction init placée dans une portée ou une méthode
   ...         paramètres
   );

void vector::Init(
  const ulong  size,
  func_name    init_func=NULL,  // fonction init placée dans une portée ou une méthode
   ...         paramètres
   );
```

Paramètres

rows

[in] Nombre de lignes.

cols

[in] Nombre de colonnes.

func_name

[in] Fonction d'initialisation.

...

[in] Paramètres de la fonction d'initialisation.

Valeur de Retour

Aucune valeur de retour.

Exemple

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
  int size_m=3, size_k=4;
  matrix  m(size_m,size_k,MatrixArange,-2.,0.1); // une matrice non initialisée de tai
  Print("matrix m \n",m);                        // puis la fonction MatrixArange avec
  matrixf m_float(5,5,MatrixArange,-2.f,0.1f);   // une matrice d'éléments de type flo
  Print("matrix m_float \n",m_float);
  vector  v(size_k,VectorArange,-10.0);          // après la création d'un vecteur, Ve
  Print("vector v \n",v);
  /*
   matrix m
   [[-2,-1.9,-1.8,-1.7]
    [-1.6,-1.5,-1.399999999999999,-1.299999999999999]
    [-1.199999999999999,-1.099999999999999,-0.999999999999992,-0.899999999999992]]
   matrix m_float
   [[-2,-1.9,-1.8,-1.6999999,-1.5999999]
    [-1.4999999,-1.3999999,-1.2999998,-1.1999998,-1.0999998]
    [-0.99999976,-0.89999974,-0.79999971,-0.69999969,-0.59999967]
    [-0.49999967,-0.39999968,-0.29999968,-0.19999969,-0.099999689]
    [3.1292439e-07,0.10000031,0.20000032,0.30000031,0.4000003]]
   vector v
   [-10,-9,-8,-7]
  */
  }
```

© 2000-2024, MetaQuotes Ltd.

Fill

Remplit une matrice ou un vecteur existant avec la valeur spécifiée.

```
void matrix::Fill(
  const double  value      // valeur de remplissage
   );

void vector::Fill(
  const double  value      // valeur de remplissage
   );
```

Paramètres

value

[in] Valeur avec laquelle les éléments de la matrice seront remplis.

Valeur de Retour

Aucune valeur de retour. La matrice est remplie en place avec la valeur spécifiée.

Exemple

```
   matrix matrix_a(2,2);
   matrix_a.Fill(10);
   Print("matrix_a\n",matrix_a);

/*
  matrix_a
  [[10,10]
   [10,10]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Manipulations matricielles et vectorielles

Ce sont des méthodes pour les opérations matricielles de base : remplissage, copie, obtention d'une partie d'une matrice, transposition, division et tri.

Il existe également plusieurs méthodes pour les opérations avec des lignes et des colonnes de matrice.

Fonction	Action
HasNan	Retourne le nombre de valeurs NaN dans une matrice/un vecteur
Transpose	Inverse ou permute les axes d'une matrice ; renvoie la matrice modifiée
TriL	Renvoie une copie d'une matrice avec des éléments au-dessus de la k-ième diagonale mis à 0. Matrice triangulaire inférieure
TriU	Renvoie une copie d'une matrice avec des éléments sous la k-ième diagonale mis à 0. Matrice triangulaire supérieure
Diag	Extrait une diagonale ou construit une matrice diagonale
Row	Renvoie un vecteur ligne. Écrit un vecteur dans la ligne spécifiée
Col	Renvoie un vecteur colonne. Écrit un vecteur dans la colonne spécifiée
Copy	Renvoie une copie de la matrice/du vecteur donné
Compare	Compare les éléments de deux matrices/vecteurs avec la précision spécifiée
CompareByDigits	Compare les éléments de deux matrices/vecteurs jusqu'à des chiffres significatifs
Flat	Permet d'adresser un élément de matrice via un indice au lieu de deux
Clip	Limiter les éléments d'une matrice/vecteur à une plage spécifiée de valeurs valides
Reshape	Modifie la forme d'une matrice sans modifier ses données
Resize	Renvoie une nouvelle matrice avec une forme et une taille modifiées
SwapRows	Échange des lignes dans une matrice
SwapCols	Echange des colonnes dans une matrice
Split	Divise une matrice en plusieurs sous-matrices
Hsplit	Divise une matrice horizontalement en plusieurs sous-matrices. Identique à Split avec axis=0
Vsplit	Divise une matrice verticalement en plusieurs sous-matrices. Identique à Split avec axis=1
ArgSort	Tri indirect d'une matrice ou d'un vecteur.

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
Sort	Tri d'une matrice ou d'un vecteur en place.

© 2000-2024, MetaQuotes Ltd.

HasNan

Retourne le nombre de valeurs NaN dans une matrice/un vecteur.

```
ulong vector::HasNan();

ulong matrix::HasNan();
```

Valeur de Retour

Le nombre d'éléments de matrice/vecteur qui contiennent une valeur NaN.

Note

Lors de la comparaison d'une paire d'éléments ayant des valeurs NaN, les méthodes Compare et CompareByDigits considèrent que ces éléments sont égaux, alors que dans le cas d'une comparaison usuelle de nombres à virgule flottante NaN != NaN.

Exemple :

```
void OnStart(void)
  {
   double x=sqrt(-1);

   Print("single: ",x==x);

   vector<double> v1={x};
   vector<double> v2={x};

   Print("vecteur: ", v1.Compare(v2,0)==0);
  }

/* Résultat :

 single: false
vecteur: true
*/
```

Voir aussi

MathClassify, Compare, CompareByDigits

© 2000-2024, MetaQuotes Ltd.

Transpose

Transposition de matrice. Inverse ou permute les axes d'une matrice ; renvoie la matrice modifiée.

```
matrix matrix::Transpose()
```

Valeur de Retour

Matrice transposée.

Un algorithme simple de transposition de matrice en MQL5 :

```
matrix MatrixTranspose(const matrix& matrix_a)
  {
   matrix matrix_c(matrix_a.Cols(),matrix_a.Rows());

   for(ulong i=0; i<matrix_c.Rows(); i++)
      for(ulong j=0; j<matrix_c.Cols(); j++)
         matrix_c[i][j]=matrix_a[j][i];

   return(matrix_c);
  }
```

Exemple en MQL5 :

```
  matrix a= {{0, 1, 2}, {3, 4, 5}};
  Print("matrix a \n", a);
  Print("a.Transpose() \n", a.Transpose());

  /*
  matrix a
  [[0,1,2]
   [3,4,5]]
  a.Transpose()
  [[0,3]
   [1,4]
   [2,5]]
  */
```

Exemple en Python :

```
import numpy as np

a = np.arange(6).reshape((2,3))
print("a \n",a)
print("np.transpose(a) \n",np.transpose(a))
```

© 2000-2024, MetaQuotes Ltd.

```
a
 [[0 1 2]
 [3 4 5]]
np.transpose(a)
 [[0 3]
 [1 4]
 [2 5]]
```

© 2000-2024, MetaQuotes Ltd.

TriL

Renvoie une copie d'une matrice avec des éléments au-dessus de la k-ième diagonale mis à 0. Matrice triangulaire inférieure.

```
matrix matrix::Tril(
  const int     ndiag=0      // indice de la diagonale
   );
```

Paramètres

ndiag=0

[in] Diagonale au-dessus de laquelle les éléments doivent être mis à 0. ndiag = 0 (valeur par défaut) est la diagonale principale, ndiag < 0 est en dessous et ndiag > 0 est au-dessus.

Valeur de Retour

Matrice avec son triangle inférieur rempli de 1 et 0 ailleurs.

Exemple en MQL5 :

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriL(-1);
   Print("matrix b \n",b);

  /*
  matrix_c
  [[0,0,0]
  [4,0,0]
  [7,8,0]
  [10,11,12]]
  */
```

Exemple en Python :

```
import numpy as np

a=np.tril([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)

[[ 0  0  0]
 [ 4  0  0]
 [ 7  8  0]
 [10 11 12]]
```

© 2000-2024, MetaQuotes Ltd.

TriU

Renvoie une copie d'une matrice avec les éléments sous la k-ième diagonale mis à 0. Matrice triangulaire supérieure.

```
matrix matrix::Triu(
  const int     ndiag=0      // indice de la diagonale
   );
```

Paramètres

ndiag=0

[in] Diagonale en dessous de laquelle les éléments doivent être mlis à 0. ndiag = 0 (valeur par défaut) est la diagonale principale, ndiag < 0 est en dessous et ndiag > 0 est au-dessus.

Exemple en MQL5 :

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriU(-1);
   Print("matrix b \n",b);

  /*
  matrix b
  [[1,2,3]
   [4,5,6]
   [0,8,9]
   [0,0,12]]
  */
```

Exemple en Python :

```
import numpy as np

a=np.triu([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)
print(a)

[[ 1  2  3]
 [ 4  5  6]
 [ 0  8  9]
 [ 0  0 12]]
```

© 2000-2024, MetaQuotes Ltd.

Diag

Extrait une diagonale ou construire une matrice diagonale.

```
vector matrix::Diag(
  const int     ndiag=0      // nombre de diagonales
   );

void matrix::Diag(
  const vector  v,           // vecteur diagonal
  const int     ndiag=0      // nombre de diagonales
   );
```

Paramètres

v

[in] Un vecteur dont les éléments seront contenus dans la diagonale correspondante (ndiag=0 est la diagonale principale).

ndiag=0

[in] Diagonale en question. La valeur par défaut est 0. Utilisez ndiag>0 pour les diagonales au-dessus de la diagonale principale et ndiag<0 pour les diagonales en dessous de la diagonale principale.

Note

Une diagonale peut être définie pour les matrices non allouées (qui n'ont pas de dimensions). Dans ce cas, une matrice nulle de la taille sera créée avec la taille correspondant à la taille du vecteur diagonal, après quoi les valeurs vectorielles seront renseignées dans la diagonale correspondante. Si la diagonale est définie sur une matrice déjà existante, les dimensions de la matrice ne changent pas et les valeurs des éléments de la matrice en dehors du vecteur diagonal ne changent pas.

Exemple

```
vector v1={1,2,3};
matrix m1;
m1.Diag(v1);
Print("m1\n",m1);
matrix m2;
m2.Diag(v1,-1);
Print("m2\n",m2);
matrix m3;
m3.Diag(v1,1);
Print("m3\n",m3);
matrix m4=matrix::Full(4,5,9);
m4.Diag(v1,1);
Print("m4\n",m4);
```

© 2000-2024, MetaQuotes Ltd.

```
 Print("diag -1 - ",m4.Diag(-1));
 Print("diag 0 - ",m4.Diag());
 Print("diag 1 - ",m4.Diag(1));

/*

m1
[[1,0,0]
[0,2,0]
[0,0,3]]
m2
[[0,0,0]
[1,0,0]
[0,2,0]
[0,0,3]]
m3
[[0,1,0,0]
[0,0,2,0]
[0,0,0,3]]
m4
[[9,1,9,9,9]
[9,9,2,9,9]
[9,9,9,3,9]
[9,9,9,9,9]]
diag -1 - [9,9,9]
diag 0 - [9,9,9,9]
diag 1 - [1,2,3,9]
*/
```

© 2000-2024, MetaQuotes Ltd.

Row

Renvoie un vecteur ligne. Écrit un vecteur dans la ligne spécifiée

```
vector matrix::Row(
  const ulong   nrow      // numéro de ligne
   );

void matrix::Row(
  const vector  v,        // vecteur ligne
  const ulong   nrow      // numéro de ligne
   );
```

Paramètres

nrow

[in] Numéro de ligne.

Valeur de Retour

Vecteur.

Note

Une ligne peut être définie pour les matrices non allouées (qui n'ont pas de dimensions). Dans ce cas, une matrice nulle sera créée avec la taille du vecteur taille x numéro de ligne + 1, après quoi les valeurs des éléments du vecteur seront renseignées dans la ligne correspondante. Si la ligne est définie sur une matrice déjà existante, les dimensions de la matrice ne changent pas et les valeurs des éléments de la matrice en dehors du vecteur ligne ne changent pas.

Exemple

```
   vector v1={1,2,3};
   matrix m1;
   m1.Row(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,7);
   m2.Row(v1,2);
   Print("m2\n",m2);

   Print("row 1 - ",m2.Row(1));
   Print("row 2 - ",m2.Row(2));

  /*
  m1
  [[0,0,0]
  [1,2,3]]
  m2
```

© 2000-2024, MetaQuotes Ltd.

```
[[7,7,7,7,7]
[7,7,7,7,7]
[1,2,3,7,7]
[7,7,7,7,7]]
row 1 - [7,7,7,7,7]
row 2 - [1,2,3,7,7]
*/
```

© 2000-2024, MetaQuotes Ltd.

Col

Renvoie un vecteur colonne. Ecrit un vecteur dans la colonne spécifiée.

```
vector matrix::Col(
  const ulong   ncol      // numéro de colonne
   );

void matrix::Col(
  const vector  v,        // vecteur colonne
  const ulong   ncol      // numéro de colonne
   );
```

Paramètres

ncol

[in] Numéro de la colonne.

Valeur de Retour

Vecteur.

Note

Une colonne peut être définie pour les matrices non allouées (qui n'ont pas de dimensions). Dans ce cas, une matrice zéro sera créée avec la taille du vecteur taille x numéro de colonne + 1, après quoi les valeurs des éléments du vecteur seront renseignées dans la colonne correspondante. Si la colonne est définie sur une matrice déjà existante, les dimensions de la matrice ne changent pas et les valeurs des éléments de la matrice en dehors du vecteur de colonne ne changent pas.

Exemple

```
   vector v1={1,2,3};
   matrix m1;
   m1.Col(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,8);
   m2.Col(v1,2);
   Print("m2\n",m2);

   Print("col 1 - ",m2.Col(1));
   Print("col 2 - ",m2.Col(2));

  /*
  m1
  [[0,1]
  [0,2]
  [0,3]]
  m2
```

© 2000-2024, MetaQuotes Ltd.

```
[[8,8,1,8,8]
[8,8,2,8,8]
[8,8,3,8,8]
[8,8,8,8,8]]
col 1 - [8,8,8,8]
col 2 - [1,2,3,8]
*/
```

© 2000-2024, MetaQuotes Ltd.

Copy

Crée une copie de la matrice/du vecteur donné.

```
bool matrix::Copy(
  const matrix&  a     // matrice copiée
   );
bool vector::Copy(
  const vector&  v     // vecteur copié
   );
```

Paramètres

v

[in] Matrice ou vecteur à copier.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Exemple en MQL5 :

```
  matrix a=matrix::Eye(3, 4);
  matrix b;
  b.Copy(a);
  matrix c=a;
  Print("matrix b \n", b);
  Print("matrix_c \n", c);

  /*
  /*
  matrix b
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  matrix_c
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  */
  */
```

Exemple en Python :

```
import numpy as np

a = np.eye(3,4)
print('a \n',a)
b = a
```

© 2000-2024, MetaQuotes Ltd.

```
print('b \n',b)
c = np.copy(a)
print('c \n',c)

a
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
b
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
c
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
```

© 2000-2024, MetaQuotes Ltd.

Compare

Compare les éléments de deux matrices/vecteurs avec la précision spécifiée.

```
ulong vector::Compare(
  const vector& vec,          // vecteur de comparaison
  const double  epsilon       // précision
   );

ulong matrix::Compare(
  const matrix& mat,          // matrice de comparaison
  const double  epsilon       // précision
   );
```

Paramètres

vector_b

[in] Vecteur de comparaison.

epsilon

[in] Précision.

Valeur de Retour

Le nombre d'éléments non concordants des matrices ou des vecteurs comparés : 0 si les matrices sont égales, supérieur à 0 sinon.

Note

Les opérateurs de comparaison == ou != exécutent une comparaison exacte élément par élément. On sait que la comparaison exacte des nombres réels est d'une utilité limitée, c'est pourquoi la méthode de comparaison epsilon a été ajoutée. Il peut arriver qu'une matrice contienne des éléments dans une plage, par exemple de 1e-20 à 1e+20. De telles matrices peuvent être traitées en utilisant une comparaison élément par élément jusqu'à des chiffres significatifs.

Exemple

```
matrix matrix_a={{10,3,2},{1,8,12},{6,5,4}};
matrix matrix_i=matrix::Identity(3,3);
matrix matrix_c=matrix_a.Inv();
matrix matrix_check=matrix_a.MatMul(matrix_c);
Print("matrix_check\n",matrix_check);

ulong errors=matrix_check.Compare(matrix::Identity(3,3),1e-15);
Print("erreurs=",errors);
```

© 2000-2024, MetaQuotes Ltd.

```
/*
matrix_check
[[1,0,0]
[4.440892098500626e-16,1,8.881784197001252e-16]
[4.440892098500626e-16,2.220446049250313e-16,0.9999999999999996]]
erreurs=0

*/
```

© 2000-2024, MetaQuotes Ltd.

CompareByDigits

Compare les éléments de deux matrices/vecteurs avec la précision des chiffres significatifs.

```
ulong vector::CompareByDigits(
  const vector& vec,          // vecteur de comparaison
  const int     digits        // nombre de chiffres significatifs
   );

ulong matrix::CompareByDigits(
  const matrix& mat,          // matrice de comparaison
  const int     digits        // nombre de chiffres significatifs
   );
```

Paramètres

vector_b

[in] Vecteur de comparaison.

digits

[in] Nombre de chiffres significatifs à utiliser dans la comparaison.

epsilon

[in] Précision de la comparaison. Si deux valeurs diffèrent en valeur absolue de moins que la précision spécifiée, elles sont considérées comme égales.

Valeur de Retour

Le nombre d'éléments non concordants des matrices ou des vecteurs comparés : 0 si les matrices sont égales, supérieur à 0 sinon.

Note

Les opérateurs de comparaison == ou != exécutent une comparaison exacte élément par élément. On sait que la comparaison exacte des nombres réels est d'une utilité limitée, c'est pourquoi la méthode de comparaison epsilon a été ajoutée. Il peut arriver qu'une matrice contienne des éléments dans une plage, par exemple de 1e-20 à 1e+20. De telles matrices peuvent être traitées en utilisant une comparaison élément par élément jusqu'à des chiffres significatifs.

Exemple

```
   int    size_m=128;
   int    size_k=256;
   matrix matrix_a(size_m,size_k);
//--- remplit la matrice
   double  value=0.0;
   for(int i=0; i<size_m; i++)
     {
      for(int j=0; j<size_k; j++)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         if(i==j)
            matrix_a[i][j]=1.0+i;
         else
           {
            value+=1.0;
            matrix_a[i][j]=value/1e+20;
           }
        }
     }
//--- récupère une autre matrice
   matrix matrix_c = matrix_a * -1;

   ulong errors_epsilon=matrix_a.Compare(matrix_c,1e-15);
   ulong errors_digits=matrix_a.CompareByDigits(matrix_c,15);

   printf("Compare une matrice %d x %d  errors_epsilon=%I64u  errors_digits=%I64u",siz


  /*
  Compare une matrice 128 x 256  errors_epsilon=128  errors_digits=32768
  */
```

© 2000-2024, MetaQuotes Ltd.

Flat

Permet d'adresser un élément de matrice via un index au lieu de deux.

```
bool matrix::Flat(
  const ulong   index,      //
  const double  value       // valeur à définir
   );

double matrix::Flat(
  const ulong   index,      //
   );
```

Paramètres

index

[in] Indice plat

value

[in] Valeur à définir par l'indice donné.

Valeur de Retour

Valeur à l'indice donné.

Note

Pour la matrice mat(3,3), on peut accéder comme suit :

- lecture : 'x=mat.Flat(4)', qui est équivalent à 'x=mat[1][1]'
- écriture : 'mat.Flat(5, 42)', qui est équivalent à 'mat[1][2]=42'

Exemple

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);
   ulong arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));
   matrix_a.Flat(arg_max,0);
   arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));


  /*
  matrix_a
  [[10,3,2]
   [1,8,12]
   [6,5,4]
   [7,11,9]]
```

© 2000-2024, MetaQuotes Ltd.

```
max_value=12.0
max_value=11.0
*/
```

© 2000-2024, MetaQuotes Ltd.

Clip

Limite les éléments d'une matrice/d'un vecteur à une plage spécifiée de valeurs valides.

```
bool matrix::Clip(
  const double  min_value,      // valeur minimum
  const double  max_value       //valeur  maximum
   );
bool vector::Clip(
  const double  min_value,      // valeur minimum
  const double  max_value       //valeur  maximum
   );
```

Paramètres

min_value

[in] Valeur minimum.

max_value

[in] Valeur maximum.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Note

La matrice (ou vecteur) est traitée sur place. Aucune copie n'est créée.

Exemple

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   bool res=matrix_a.Clip(4,8);
   Print("matrix_a\n",matrix_a);

  /*
  matrix_a
  [[4,4,4]
   [4,5,6]
   [7,8,8]
   [8,8,8]]
  */
```

© 2000-2024, MetaQuotes Ltd.

Reshape

Modifie la forme d'une matrice sans modifier ses données.

```
void  Reshape(
  const ulong  rows,      // nouveau nombre de lignes
  const ulong  cols       // nouveau nombre de colonnes
   );
```

Paramètres

rows

[in] Nouveau nombre de lignes.

cols

[in] Nouveau nombre de colonnes.

Note

La matrice est traitée sur place. Aucune copie n'est créée. N'importe quelle taille peut être spécifiée, c'est-à-dire rows_new*cols_new!=rows_old*cols_old. Lorsque le buffer de la matrice est augmenté, les valeurs supplémentaires sont indéfinies.

Exemple

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};

   Print("matrix_a\n",matrix_a);
   matrix_a.Reshape(2,6);
   Print("Reshape(2,6)\n",matrix_a);
   matrix_a.Reshape(3,5);
   Print("Reshape(3,5)\n",matrix_a);
   matrix_a.Reshape(2,4);
   Print("Reshape(2,4)\n",matrix_a);

  /*
  matrix_a
  [[1,2,3]
   [4,5,6]
   [7,8,9]
   [10,11,12]]
  Reshape(2,6)
  [[1,2,3,4,5,6]
   [7,8,9,10,11,12]]
  Reshape(3,5)
  [[1,2,3,4,5]
   [6,7,8,9,10]
   [11,12,0,3,0]]
  Reshape(2,4)
  [[1,2,3,4]
```

© 2000-2024, MetaQuotes Ltd.

```
  [5,6,7,8]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Resize

Retourne une nouvelle matrice avec une forme et une taille modifiées.

```
bool matrix::Resize(
  const ulong  rows,      //
  const ulong  cols,      // nouveau nombre de colonnes
  const ulong  reserve=0 // quantité d'éléments de réserve
   );

bool vector::Resize(
  const ulong  size,      // nouvelle taille
  const ulong  reserve=0 // quantité d'éléments de réserve
   );
```

Paramètres

rows

[in] Nouveau nombre de lignes.

cols

[in] Nouveau nombre de colonnes.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Note

La matrice (ou vecteur) est traitée sur place. Aucune copie n'est créée. N'importe quelle taille peut être spécifiée, c'est-à-dire rows_new*cols_new!=rows_old*cols_old. Contrairement à Reshape, la matrice est traitée ligne par ligne. Lorsque vous augmentez le nombre de colonnes, les valeurs des colonnes supplémentaires sont indéfinies. Lors de l'augmentation du nombre de lignes, les valeurs des éléments dans les nouvelles lignes sont indéfinies. Lorsque le nombre de colonnes est réduit, chaque ligne de la matrice est tronquée.

Exemple

```
matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
Print("matrix_a\n",matrix_a);
matrix_a.Resize(2,6);
Print("Resize(2,6)\n",matrix_a);
matrix_a.Resize(3,5);
Print("Resize(3,5)\n",matrix_a);
matrix_a.Resize(2,4);
Print("Resize(2,4)\n",matrix_a);

/*
matrix_a
```

© 2000-2024, MetaQuotes Ltd.

```
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
Resize(2,6)
[[1,2,3,4,5,6]
 [4,5,6,10,11,12]]
Resize(3,5)
[[1,2,3,4,5]
 [4,5,6,10,11]
 [11,12,3,8,8]]
Resize(2,4)
[[1,2,3,4]
 [4,5,6,10]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Set

Définit la valeur de l'élément d'un vecteur en spécifiant son indice.

```
bool vector::Set(
  ulong   index,      // indice de l'élément
  double  value       // valeur
   );
```

Paramètres

index

[in] Indice de l'élément pour lequel on veut changer la valeur.

value

[in] Valeur.

Valeur de Retour

Retourne true en cas de succès, false sinon.

Note

La méthode Set fait la même chose que l'attribution d'une valeur à l'aide de crochets, à savoir : *vecteur[index]=valeur*. La méthode a été ajoutée pour simplifier le transfert d'un code depuis des langages où ce type de notation est utilisé. L'exemple ci-dessous montre les deux options pour remplir le vecteur avec des valeurs selon l'index spécifié.

Exemple :

```
void OnStart()
  {
//---
   vector v1(10, VectorAssignValues);
   Print("v1 = ", v1);

   vector v2(10, VectorSetValues);
   Print("v2 = ", v2);
  }
 /* Résultat
  v1 = [1,2,4,8,16,32,64,128,256,512]
  v2 = [1,2,4,8,16,32,64,128,256,512]
  */
//+------------------------------------------------------------------------+
//| Remplis un vecteur avec les puissances d'un nombre via des affectations |
//+------------------------------------------------------------------------+
void VectorAssignValues(vector& v, double initial=1)
  {
   double value=initial;
```

© 2000-2024, MetaQuotes Ltd.

```
   for(ulong k=0; k<v.Size(); k++)
     {
      v[k]=value;
      value*=2;
     }
  }
//+------------------------------------------------------------------------+
//| Remplis un vecteur avec les puissances d'un nombre via la méthode Set |
//+------------------------------------------------------------------------+
void VectorSetValues(vector& v, double initial=1)
  {
   double value=initial;
   for(ulong k=0; k<v.Size(); k++)
     {
      v.Set(k, value);
      value*=2;
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

SwapRows

Échange les lignes d'une matrice.

```
bool matrix::SwapRows(
  const ulong  row1,      // indice de la première ligne
  const ulong  row2       // indice de la deuxième ligne
   );
```

Paramètres

row1

[in] Indice de la première ligne.

row2

[in] Indice de la deuxième ligne.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Exemple

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapRows(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapRows(0,3);
   matrix matrix_c1=matrix_p.MatMul(matrix_a);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  matrix_c1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  */
```

© 2000-2024, MetaQuotes Ltd.

SwapCols

Permute les colonnes d'une matrice.

```
bool matrix::SwapCols(
  const ulong  row1,      // indice de la première colonne
  const ulong  row2       // indice de la deuxième colonne
   );
```

Paramètres

col1

[in] Indice de la première colonne.

col2

[in] Indice de la deuxième colonne.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Exemple

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapCols(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapCols(0,3);
   matrix matrix_c1=matrix_a.MatMul(matrix_p);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  matrix_c1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  */
```

© 2000-2024, MetaQuotes Ltd.

Split

Divise une matrice en plusieurs sous-matrices.

```
bool matrix::Split(
  const ulong  parts,      // nombre de sous-matrices
  const int    axis,       // axe
  matrix&      splitted[]  // tableau de sous-matrices résultantes
   );

void matrix::Split(
  const ulong& parts[],    // tailles des sous-matrices
  const int    axis,       // axe
  matrix&      splitted[]  // tableau de sous-matrices résultantes
   );
```

Paramètres

parts

[in] Le nombre de sous-matrices dans lesquelles diviser la matrice.

axis

[in] Axe. 0 - axe horizontal, 1 - axe vertical.

splitted

[out] Tableau des sous-matrices résultantes.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Note

Si le nombre de sous-matrices est spécifié, alors des sous-matrices de même taille sont obtenues. Cela signifie que la taille de la matrice (0 - le nombre de lignes, 1 - le nombre de colonnes) doit être divisible par des "parties" sans reste. Des sous-matrices de tailles différentes peuvent être obtenues en utilisant un tableau de tailles de sous-matrices. Les éléments du tableau de taille sont utilisés jusqu'à ce que la matrice entière soit divisée. Si le tableau des tailles est terminé et que la matrice n'a pas encore été complètement divisée, le reste non divisé sera la dernière sous-matrice.

Exemple

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
```

© 2000-2024, MetaQuotes Ltd.

```
 ulong  parts[]={2,2};

 bool res=matrix_a.Split(2,0,splitted);
 Print(res,"  ",GetLastError());
 ResetLastError();
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Split(2,1,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Split(parts,0,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

/*
false  4003
true  0
splitted 0
[[1,2,3]
 [7,8,9]
 [13,14,15]
 [19,20,21]
 [25,26,27]]
splitted 1
[[4,5,6]
 [10,11,12]
 [16,17,18]
 [22,23,24]
 [28,29,30]]
true  0
splitted 0
[[1,2,3,4,5,6]
 [7,8,9,10,11,12]]
splitted 1
[[13,14,15,16,17,18]
 [19,20,21,22,23,24]]
splitted 2
[[25,26,27,28,29,30]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Hsplit

Divise une matrice horizontalement en plusieurs sous-matrices. Similaire à Split avec axis=0

```
bool matrix::Hsplit(
  const ulong  parts,      // nombre de sous-matrices
  matrix&      splitted[]  // tableau de sous-matrices résultantes
   );

void matrix::Hsplit(
  const ulong& parts[],    // tailles des sous-matrices
  matrix&      splitted[]  // tableau de sous-matrices résultantes
   );
```

Paramètres

parts

[in] Le nombre de sous-matrices dans lesquelles diviser la matrice.

splitted

[out] Tableau des sous-matrices résultantes.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Note

Si le nombre de sous-matrices est spécifié, alors des sous-matrices de même taille sont obtenues. Cela signifie que le nombre de lignes doit être divisible par des "parties" sans reste. Des sous-matrices de tailles différentes peuvent être obtenues en utilisant un tableau de tailles de sous-matrices. Les éléments du tableau de taille sont utilisés jusqu'à ce que la matrice entière soit divisée. Si le tableau des tailles est terminé et que la matrice n'a pas encore été complètement divisée, le reste non divisé sera la dernière sous-matrice.

Exemple

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
   ulong  parts[]={2,4};

   bool res=matrix_a.Hsplit(2,splitted);
   Print(res,"  ",GetLastError());
   ResetLastError();
   for(uint i=0; i<splitted.Size(); i++)
```

© 2000-2024, MetaQuotes Ltd.

```
      Print("splitted ",i,"\n",splitted[i]);

   res=matrix_a.Hsplit(5,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   res=matrix_a.Hsplit(parts,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);


/*
false  4003
true  0
splitted 0
[[1,2,3,4,5,6]]
splitted 1
[[7,8,9,10,11,12]]
splitted 2
[[13,14,15,16,17,18]]
splitted 3
[[19,20,21,22,23,24]]
splitted 4
[[25,26,27,28,29,30]]
true  0
splitted 0
[[1,2,3,4,5,6]
[7,8,9,10,11,12]]
splitted 1
[[13,14,15,16,17,18]
[19,20,21,22,23,24]
[25,26,27,28,29,30]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Vsplit

Divise verticalement une matrice en plusieurs sous-matrices. Identique à Split avec axis=1

```
bool matrix::Vsplit(
  const ulong  parts,      // nombre de sous-matrices
  matrix&      splitted[]  // tableau de sous-matrices résultantes
   );

void matrix::Vsplit(
  const ulong& parts[],    // tailles des sous-matrices
  matrix&      splitted[]  // tableau de sous-matrices résultantes
   );
```

Paramètres

parts

[in] Le nombre de sous-matrices dans lesquelles diviser la matrice.

splitted

[out] Tableau des sous-matrices résultantes.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Note

Si le nombre de sous-matrices est spécifié, alors des sous-matrices de même taille sont obtenues. Cela signifie que le nombre de colonnes doit être divisible par des "parties" sans reste. Des sous-matrices de tailles différentes peuvent être obtenues en utilisant un tableau de tailles de sous-matrices. Les éléments du tableau de taille sont utilisés jusqu'à ce que la matrice entière soit divisée. Si le tableau des tailles est terminé et que la matrice n'a pas encore été complètement divisée, le reste non divisé sera la dernière sous-matrice.

Exemple

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18}};
   matrix splitted[];
   ulong  parts[]={2,3};

   matrix_a.Vsplit(2,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   matrix_a.Vsplit(3,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);
```

© 2000-2024, MetaQuotes Ltd.

```
 matrix_a.Vsplit(parts,splitted);
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);


/*
   splitted 0
   [[1,2,3]
    [7,8,9]
    [13,14,15]]
   splitted 1
   [[4,5,6]
    [10,11,12]
    [16,17,18]]

   splitted 0
   [[1,2]
    [7,8]
    [13,14]]
   splitted 1
   [[3,4]
    [9,10]
    [15,16]]
   splitted 2
   [[5,6]
    [11,12]
    [17,18]]

   splitted 0
   [[1,2]
    [7,8]
    [13,14]]
   splitted 1
   [[3,4,5]
    [9,10,11]
    [15,16,17]]
   splitted 2
   [[6]
    [12]
    [18]]

*/
```

© 2000-2024, MetaQuotes Ltd.

ArgSort

Tri indirect d'une matrice ou d'un vecteur.

```
vector vector::Sort(
  func_name  compare_func=NULL,  // fonction de comparaison
  T          context             // paramètre de la fonction de tri personnalisé
   );

matrix matrix::Sort(
  func_name  compare_func=NULL   // fonction de comparaison
  T          context             // paramètre de la fonction de tri personnalisé
   );

matrix matrix::Sort(
  const int  axis,               // axe de tri
  func_name  compare_func=NULL   // fonction de comparaison
  T          context             // paramètre de la fonction de tri personnalisé
   );
```

Paramètres

axis

[dans] L'axe selon lequel trier : 0 horizontal, 1 vertical.

func_name

[in] Comparateur. Vous pouvez spécifier l'une des valeurs de l'énumération ENUM_SORT_MODE ou votre propre fonction de comparaison. Si aucune fonction n'est spécifiée, le tri croissant est utilisé.

Une fonction de comparaison personnalisée peut être de deux types :

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Ici, T est le type de matrice ou de vecteur, et TContex est le type de la variable 'context' qui est passée comme paramètre supplémentaire à la méthode Sort.

context

[in] Paramètre facultatif supplémentaire pouvant être passé à une fonction de tri personnalisée.

Valeur de Retour

Vecteur ou matrice avec les indices des éléments triés. Par exemple, le résultat [4,2,0,1,3] indique qu'il doit y avoir un élément avec l'index 4 en position zéro, un élément avec l'index 2 en première position, et ainsi de suite.

© 2000-2024, MetaQuotes Ltd.

Sort

Tri d'une matrice ou d'un vecteur en place.

```
void vector::Sort(
  func_name  compare_func=NULL,  // fonction de comparaison
  T          context             // paramètre de la fonction de tri personnalisé
   );

void matrix::Sort(
  func_name  compare_func=NULL   // fonction de comparaison
  T          context             // paramètre de la fonction de tri personnalisé
   );

void matrix::Sort(
  const int  axis,               // axe de tri
  func_name  compare_func=NULL   // fonction de comparaison
  T          context             // paramètre de la fonction de tri personnalisé
   );
```

Paramètres

axis

[dans] L'axe selon lequel trier : 0 horizontal, 1 vertical.

func_name

[in] Comparateur. Vous pouvez spécifier l'une des valeurs de l'énumération ENUM_SORT_MODE ou votre propre fonction de comparaison. Si aucune fonction n'est spécifiée, le tri croissant est utilisé.

Une fonction de comparaison personnalisée peut être de deux types :

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Ici, T est le type de matrice ou de vecteur, et TContex est le type de la variable 'context' qui est passée comme paramètre supplémentaire à la méthode Sort.

context

[in] Paramètre facultatif supplémentaire pouvant être passé à une fonction de tri personnalisée.

Valeur de Retour

Aucune. Le tri est effectué en place, c'est-à-dire qu'il est appliqué aux données de la matrice/vecteur pour laquelle la méthode Sort est appelée.

Exemple

```
//+------------------------------------------------------------------+
//| Fonction de tri                                                  |
//+------------------------------------------------------------------+
int MyDoubleComparator(double x1,double x2,int sort_mode=0)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   int res=x1<x2 ? -1 : (x1>x2 ? 1 : 0);
   return(sort_mode==0 ? res : -res);
  }
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- remplit le vecteur
   vector v(100);
   //--- tri ascendant
   v.Sort(MyDoubleComparator);   // un paramètre supplémentaire avec la valeur '0' pa
   Print(v);
   // tri descendant
   v.Sort(MyDoubleComparator,1); // le paramètre supplémentaire avec la valeur '1' est
   Print(v);
  }
```

© 2000-2024, MetaQuotes Ltd.

Opérations mathématiques avec des matrices et des vecteurs

Les opérations mathématiques, y compris l'addition, la soustraction, la multiplication et la division, peuvent être effectuées sur des matrices et des vecteurs élément par élément.

Les fonctions mathématiques ont été conçues à l'origine pour effectuer des opérations pertinentes sur des valeurs scalaires. La plupart des fonctions peuvent être appliquées aux matrices et vecteurs. Celles-ci incluent MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh et MathTanh . De telles opérations impliquent un traitement élément par élément des matrices et des vecteurs. Exemple

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
S    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

Pour MathMod et MathPow, le deuxième élément peut être soit un scalaire, soit une matrice/vecteur de la taille appropriée.

L'exemple suivant montre comment calculer l'écart type en appliquant des fonctions mathématiques à un vecteur.

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- utilise la fonction d'initialisation pour remplir le vecteur
  vector r(10, ArrayRandom); // tableau de nombres aléatoires de 0 à 1
//--- calcule la valeur moyenne
  double avr=r.Mean();       // valeur moyenne du tableau
  vector d=r-avr;            // calcule un tableau d'écarts par rapport à la moyenne
  Print("avr(r)=", avr);
  Print("r=", r);
  Print("d=", d);
  vector s2=MathPow(d, 2);   // tableau des écarts au carré
  double sum=s2.Sum();       // somme des écarts au carré
//--- calcule l'écart type dans deux méthodes différentes
  double std=MathSqrt(sum/r.Size());
  Print(" std(r)=", std);
```

© 2000-2024, MetaQuotes Ltd.

```
  Print("r.Std()=", r.Std());
 }
/*
  avr(r)=0.5300302133243813
  r=[0.8346201971495713,0.8031556138798182,0.6696676534318063,0.05386516922513505,0.54
  d=[0.30458998382519,0.2731254005554369,0.1396374401074251,-0.4761650440992462,0.0190
   std(r)=0.2838269732183663
  r.Std()=0.2838269732183663
*/
//+------------------------------------------------------------------+
//| Remplissage d'un vecteur avec des valeurs aléatoires             |
//+------------------------------------------------------------------+
void ArrayRandom(vector& v)
 {
  for(ulong i=0; i<v.Size(); i++)
    v[i]=double(MathRand())/32767.;
 }
```

© 2000-2024, MetaQuotes Ltd.

Opérations mathématiques

Les opérations mathématiques, y compris l'addition, la soustraction, la multiplication et la division, peuvent être effectuées sur des matrices et des vecteurs élément par élément.

Les deux matrices ou les deux vecteurs doivent être du même type et avoir les mêmes dimensions. Chaque élément de la matrice opère sur l'élément correspondant de la seconde matrice.

Vous pouvez également utiliser un scalaire du type approprié (double, flottant ou complexe) comme deuxième terme (multiplicateur, sous-traitant ou diviseur). Dans ce cas, chaque membre de la matrice ou du vecteur fonctionnera sur le scalaire spécifié.

```
  matrix matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
  matrix matrix_b={{1,2,3},{4,5,6}};

  matrix matrix_c1=matrix_a+matrix_b;
  matrix matrix_c2=matrix_b-matrix_a;
  matrix matrix_c3=matrix_a*matrix_b;   // Produit Hadamard, à ne pas confondre avec 
  matrix matrix_c4=matrix_b/matrix_a;

  matrix_c1=matrix_a+1;
  matrix_c2=matrix_b-double_value;
  matrix_c3=matrix_a*M_PI;
  matrix_c4=matrix_b/0.1;

//--- les opérations en place sont possibles
  matrix_a+=matrix_b;
  matrix_a/=2;
```

Les mêmes opérations sont disponibles pour les vecteurs.

© 2000-2024, MetaQuotes Ltd.

Fonctions mathématiques

Les fonctions mathématiques suivantes peuvent être appliquées aux matrices et aux vecteurs : MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, MathTanh. De telles opérations impliquent un traitement élément par élément des matrices et des vecteurs.

Pour MathMod et MathPow, le deuxième élément peut être soit un scalaire, soit une matrice/un vecteur de la taille appropriée.

```
   matrix<T> mat1(128,128);
   matrix<T> mat3(mat1.Rows(),mat1.Cols());
   ulong     n,size=mat1.Rows()*mat1.Cols();
...
   mat2=MathPow(mat1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),(T)1.9);
      if(res!=mat2.Flat(n))
         errors++;
     }
   mat2=MathPow(mat1,mat3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),mat3.Flat(n));
      if(res!=mat2.Flat(n))
         errors++;
     }
...
   vector<T> vec1(16384);
   vector<T> vec3(vec1.Size());
   ulong     n,size=vec1.Size();
...
   vec2=MathPow(vec1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],(T)1.9);
      if(res!=vec2[n])
         errors++;
     }
   vec2=MathPow(vec1,vec3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],vec3[n]);
      if(res!=vec2[n])
         errors++;
     }
```

© 2000-2024, MetaQuotes Ltd.

Produits matriciels et vectoriels

Les calculs de produits matriciels et vectoriels incluent :

- multiplication matricielle
- multiplication vectorielle
- calcul de la matrice de covariance
- calcul de la corrélation croisée de deux vecteurs
- calcul de la convolution de deux vecteurs
- calcul du coefficient de corrélation

Fonction	Action
MatMul	Produit matriciel de deux matrices
GeMM	Multiplication matricielle générale (GeMM)
Power	Élevation d'une matrice carrée à la puissance entière
Dot	Produit scalaire de deux vecteurs
Kron	Calcul du produit de Kronecker de deux matrices, matrice et vecteur, vecteur et matrice ou deux vecteurs
Inner	Produit interne de deux matrices
Outer	Produit extérieur de deux matrices ou de deux vecteurs
CorrCoef	Calcul du coefficient de corrélation de Pearson (coefficient de corrélation linéaire)
Cov	Calcul de la matrice de covariance
Correlate	Calcul de la corrélation croisée de deux vecteurs
Convolve	Renvoie la convolution discrète et linéaire de deux vecteurs

© 2000-2024, MetaQuotes Ltd.

MatMul

La méthode MatMul, qui permet la multiplication de matrices et de vecteurs, comporte plusieurs surcharges.

Multiplication d'une matrice par une matrice : `matrix[M][K] * matrix[K][N] = matrix[M][N]`

```
matrix  matrix::MatMul(
  const matrix&  b      // deuxième matrice
   );
```

Multiplication d'un vecteur par une matrice : `horizontal vector[K] * matrix[K][N] = horizontal vector[N]`

```
vector  vector::MatMul(
  const matrix&  b      // matrice
   );
```

Multiplication d'une matrice par un vecteur : `matrix[M][K] * vertical vector[K] = vertical vector[M]`

```
vector  matrix::MatMul(
  const vector&  b      // vecteur
   );
```

Multiplication d'un vecteur par un scalaire : `horizontal vector * vertical vector = dot value`

```
scalar  vector::MatMul(
  const vector&  b      // deuxième vecteur
   );
```

Paramètres

b

[in] Matrice ou vecteur.

Valeur de Retour

Matrice, vecteur ou scalaire, selon la méthode utilisée.

Note

Les matrices doivent être compatibles pour la multiplication, c'est-à-dire que le nombre de colonnes dans la première matrice doit être égal au nombre de lignes dans la deuxième matrice. La multiplication matricielle est non commutative : le résultat de la multiplication de la première matrice par la seconde n'est pas égal au résultat de la multiplication de la seconde matrice par la première dans le cas général.

Le produit matriciel est constitué de toutes les combinaisons possibles de produits scalaires des vecteurs lignes de la première matrice et des vecteurs colonnes de la seconde matrice.

Dans la multiplication scalaire, les vecteurs doivent avoir la même longueur.

© 2000-2024, MetaQuotes Ltd.

Lors de la multiplication d'un vecteur et d'une matrice, la longueur du vecteur doit correspondre exactement au nombre de colonnes de la matrice.

Algorithme naïf de multiplication matricielle en MQL5 :

```
matrix MatrixProduct(const matrix& matrix_a, const matrix& matrix_b)
  {
   matrix matrix_c;

   if(matrix_a.Cols()!=matrix_b.Rows())
      return(matrix_c);

   ulong M=matrix_a.Rows();
   ulong K=matrix_a.Cols();
   ulong N=matrix_b.Cols();
   matrix_c=matrix::Zeros(M,N);

   for(ulong m=0; m<M; m++)
      for(ulong k=0; k<K; k++)
         for(ulong n=0; n<N; n++)
            matrix_c[m][n]+=matrix_a[m][k]*matrix_b[k][n];

   return(matrix_c);
  }
```

Exemple de multiplication matricielle

```
   matrix a={{1, 0, 0},
             {0, 1, 0}};
   matrix b={{4, 1},
             {2, 2},
             {1, 3}};
   matrix c1=a.MatMul(b);
   matrix c2=b.MatMul(a);
   Print("c1 = \n", c1);
   Print("c2 = \n", c2);
/*
   c1 =
   [[4,1]
    [2,2]]
   c2 =
   [[4,1,0]
    [2,2,0]
    [1,3,0]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Un exemple de multiplication d'un vecteur horizontal par une matrice

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée une matrice 3x5
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   vector v3 = {1, 2, 3};
   Print("Produit du vecteur horizontal v et de la matrice m[3,5]");
   Print("Sur la gauche, vecteur v3 = ", v3);
   Print("Sur la droite, matrice m35 = \n", m35);
   Print("v3.MatMul(m35) = vecteur horizontal v[5] \n", v3.MatMul(m35));

  /* Résultat
    Produit du vecteur horizontal v3 et de la matrice m[3,5]
    Sur la gauche, vecteur v3 = [1,2,3]
    Sur la droite, matrice m35 =
    [[0,1,2,3,4]
     [5,6,7,8,9]
     [10,11,12,13,14]]
    v3.MatMul(m35) = vecteur horizontal v[5]
    [40,46,52,58,64]
   */
  }
//+------------------------------------------------------------------+
//|  Remplit la matrice avec des valeurs croissantes                 |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

Un exemple de comment multiplier une matrice par un vecteur vertical

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée une matrice 3x5
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   Print("Produit de la matrice m[3,5] et du vecteur vertical v[5]");
   vector v5 = {1,2,3,4,5};
   Print("Sur la gauche, m35 = \n",m35);
   Print("Sur la droite, v5 = ",v5);
   Print("m35.MatMul(v5) = vecteur vertical v[3] \n",m35.MatMul(v5));

  /* Résultat
   Produit de la matrice m[3,5] et du vecteur vertical v[5]
   Sur la gauche, m35 =
   [[0,1,2,3,4]
    [5,6,7,8,9]
    [10,11,12,13,14]]
   Sur la droite, v5 = [1,2,3,4,5]
   m35.MatMul(v5) = vecteur vertical v[3]
   [40,115,190]
   */
  }
//+------------------------------------------------------------------+
//|  Remplit la matrice avec des valeurs croissantes                 |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

Un exemple de produit scalaire (point) de vecteurs

```
void OnStart()
  {
//--- produit scalaire d'un vecteur horizontal et d'un vecteur vertical
   vector a= {1, 2, 3};  // vecteur horizontal
   vector b= {4, 5, 6};  // vecteur vertical
   Print("a = ", a);
   Print("b = ", b);
   Print("1) a.MatMul(b) = ", a.MatMul(b));
   //--- voyez que la méthode Dot génère le même résultat
   Print("2) a.Dot(b) = ", a.Dot(b));

  /* Résultat
   a = [1,2,3]
   b = [4,5,6]
   1) a.MatMul(b) = 32.0
   2) a.Dot(b) = 32.0
    */
  }
```

Voir aussi

Dot, GeMM

© 2000-2024, MetaQuotes Ltd.

GeMM

La méthode GeMM (General Matrix Multiply) implémente la multiplication générale de deux matrices. L'opération est définie comme C ← α A B + β C , où les matrices A et B peuvent être éventuellement transposées. Avec une multiplication normale des matrices AB (MatMul), le scalaire *alpha* est supposé égal à 1 et *bêta* égal à zéro.

La principale différence entre GeMM et MatMul en termes d'efficacité est que MatMul crée toujours un nouvel objet matrice/vecteur, tandis que GeMM fonctionne avec un objet matrice existant et ne le recrée pas. Par conséquent, lorsque vous utilisez GeMM et pré-allouez de la mémoire pour la matrice correspondante, alors, tout en travaillant avec les mêmes tailles de matrice, il n'y aura pas de réallocation de mémoire. Cela peut être un avantage important de GeMM pour le calcul en masse, par exemple, lors de l'exécution d'optimisations dans un testeur de stratégie ou lors de la formation d'un réseau de neurones.

De la même façon que MatMul, GeMM a également 4 surcharges. Cependant, la sémantique de la quatrième surcharge a été modifiée afin de permettre la multiplication d'un vecteur vertical par un vecteur horizontal.

Dans un objet matrice/vecteur existant, il n'est pas nécessaire de pré-allouer de la mémoire. La mémoire sera allouée et remplie de 0 au premier appel GeMM.

Multiplication d'une matrice par une matrice : `matrix C[M][N] = α * ( matrix A[M][K] * matrix B[K][N]) + β * matrix C[M][N]`

```
bool  matrix::GeMM(
  const matrix &A,     // première matrice
  const matrix &B,     // deuxième matrice
  double alpha,        // multiplicateur alpha pour le produit AB
  double beta,         // multiplicateur beta pour la matrice C
  uint    flags         // une combinaison de valeurs ENUM_GEMM (OR au niveau du bit ),
   );
```

Multiplication d'un vecteur par une matrice : `vector C[N] = α * ( vector A[K] * matrix B[K][N]) + β * vector C[N]`

```
bool  vector::GeMM(
  const vector &A,     // vecteur horizontal
  const matrix &B,     // matrice
  double alpha,        // multiplicateur alpha pour le produit AB
  double beta,         // multiplicateur beta pour le vecteur C
  uint   flags         // ENUM_GEMM valeur d'énumération qui détermine si la matrice A
   );
```

Multiplication d'une matrice par un vecteur : `vector C[M] = α * ( matrix A[M][K] * vector B[K] * ) + β * vector C[M]`

```
bool  vector::GeMM(
  const matrix &A,     // matrice
  const vector &B,     // vecteur vertical
  double alpha,        // multiplicateur alpha pour le produit AB
  double beta,         // multiplicateur beta pour le vecteur C
```

© 2000-2024, MetaQuotes Ltd.

```
  uint   flags        // ENUM_GEMM valeur d'énumération qui détermine si la matrice B
   );
```

Multiplication d'un vecteur par un vecteur : `matrix C[M][N] = α * ( vector A[M] * vector B[N] * ) + β * matrix C[M][N]`. Cette surcharge renvoie une matrice, contrairement à MatMul où elle renvoie un scalaire.

```
bool  matrix::GeMM(
  const vector &A,     // premier vecteur
  const vector &B,     // deuxième vecteur
  double alpha,        // multiplicateur alpha pour le produit AB
  double beta,         // beta pour la matrice C
  uint   flags         // ENUM_GEMM valeur d'énumération qui détermine si la matrice C
   );
```

Paramètres

A

[in] Matrice ou vecteur.

B

[in] Matrice ou vecteur.

alpha

[in] Multiplicateur alpha pour le produit AB.

beta

[in] Multiplicateur beta pour la matrice C résultante.

flags

[in] La valeur d'énumération ENUM_GEMM qui détermine si les matrices A, B et C sont transposées.

Valeur de Retour

Renvoie true en cas de succès ou false sinon.

ENUM_GEMM

Valeurs de l'énumération pour la méthode GeMM.

ID	Description
TRANSP_A	Utilise la matrice transposée A
TRANSP_B	Utilise la matrice B transposée
TRANSP_C	Utilise la matrice transposée C

Note

© 2000-2024, MetaQuotes Ltd.

Les matrices et les vecteurs de types float, double et complex peuvent être utilisés comme paramètres A et B. Les variantes de modèle de la méthode GeMM sont les suivantes :

```
bool matrix<T>::GeMM(const matrix<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags)
bool matrix<T>::GeMM(const vector<T> &A,const vector<T> &B,T alpha,T beta,ulong flags)

bool vector<T>::GeMM(const vector<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags)
bool vector<T>::GeMM(const matrix<T> &A,const vector<T> &B,T alpha,T beta,ulong flags)
```

Fondamentalement, la fonction générale de multiplication matricielle est décrite comme suit :

```
C[m,n] = α *Sum(A[m,k]*B[k,n]) + β*C[m,n]
```

avec les tailles suivantes : la matrice A est M x K, la matrice B est K x N et la matrice C est M x N.

Les matrices doivent donc être compatibles pour la multiplication, c'est-à-dire que le nombre de colonnes dans la première matrice doit être égal au nombre de lignes dans la deuxième matrice. La multiplication matricielle est non commutative : le résultat de la multiplication de la première matrice par la seconde n'est pas égal au résultat de la multiplication de la seconde matrice par la première dans le cas général.

Exemple :

```
void OnStart()
  {
   vector vector_a= {1, 2, 3, 4, 5};
   vector vector_b= {4, 3, 2, 1};
   matrix matrix_c;
//--- calcule GeMM pour deux vecteurs
   matrix_c.GeMM(vector_a, vector_b, 1, 0);
   Print("matrix_c:\n ", matrix_c, "\n");
   /*
   matrix_c:
    [[4,3,2,1]
    [8,6,4,2]
    [12,9,6,3]
    [16,12,8,4]
    [20,15,10,5]]
   */
//--- crée des matrices sous forme de vecteurs
   matrix matrix_a(5, 1);
   matrix matrix_b(1, 4);
   matrix_a.Col(vector_a, 0);
   matrix_b.Row(vector_b, 0);
   Print("matrix_a:\n ", matrix_a);
   Print("matrix_b:\n ", matrix_b);
   /*
```

© 2000-2024, MetaQuotes Ltd.

```
   matrix_a:
   [[1]
   [2]
   [3]
   [4]
   [5]]
   matrix_b:
   [[4,3,2,1]]
   */
//-- calcule GeMM pour deux matrices et obtient le même résultat
   matrix_c.GeMM(matrix_a, matrix_b, 1, 0);
   Print("matrix_c:\n ", matrix_c);
   /*
   matrix_c:
    [[4,3,2,1]
    [8,6,4,2]
    [12,9,6,3]
    [16,12,8,4]
    [20,15,10,5]]
   */
  }
```

Voir aussi

MatMul

© 2000-2024, MetaQuotes Ltd.

Power

Éleve une matrice carrée à la puissance entière.

```
matrix matrix::Power(
  const int  power      // puissance
   );
```

Paramètres

power

[in] L'exposant peut être n'importe quel nombre entier, positif, négatif ou nul.

Valeur de Retour

Matrice.

Note

La matrice résultante a la même taille que la matrice d'origine. Pour une élévation à la puissance 0, la matrice identité est renvoyée. La puissance positive n signifie que la matrice d'origine est multipliée n fois par elle-même. La puissance négative -n signifie que la matrice d'origine est d'abord inversée, puis la matrice inversée est multipliée par elle-même n fois.

Un algorithme simple pour élever une matrice à une puissance en MQL5 :

```
bool MatrixPower(matrix& c, const matrix& a, const int power)
  {
//--- la matrice doit être carrée
   if(a.Rows()!=a.Cols())
      return(false);
//--- la taille de la matrice résultante est exactement la même
   ulong  rows=a.Rows();
   ulong  cols=a.Cols();
   matrix result(rows,cols);
//--- lorsqu'elle est élevée à zéro, renvoie la matrice d'identité
   if(power==0)
      result.Identity();
   else
     {
      //--- pour une élévation négative, inversez d'abord la matrice
      if(power<0)
        {
         matrix inverted=a.Inv();
         result=inverted;
         for(int i=-1; i>power; i--)
            result=result.MatMul(inverted);
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      else
        {
         result=a;
         for(int i=1; i<power; i++)
            result=result.MatMul(a);
        }
     }
//---
   c=result;
   return(true);
  }
```

Exemple en MQL5 :

```
  matrix i= {{0, 1}, {-1, 0}};
  Print("i:\n", i);

  Print("i.Power(3):\n", i.Power(3));

  Print("i.Power(0):\n", i.Power(0));

  Print("i.Power(-3):\n", i.Power(-3));

  /*
  i:
  [[0,1]
   [-1,0]]

  i.Power(3):
  [[0,-1]
   [1,0]]

  i.Power(0):
  [[1,0]
   [0,1]]

  i.Power(-3):
  [[0, -1]
   [1,0]]
  */
```

Exemple en Python :

```
import numpy as np
from numpy.linalg import matrix_power

# matrix equiv. of the imaginary unit
```

© 2000-2024, MetaQuotes Ltd.

```
i = np.array([[0, 1], [-1, 0]])
print("i:\n",i)

# should = -i
print("matrix_power(i, 3) :\n",matrix_power(i, 3) )

print("matrix_power(i, 0):\n",matrix_power(i, 0))

# should = 1/(-i) = i, but w/ f.p. elements
print("matrix_power(i, -3):\n",matrix_power(i, -3))

i:
 [[ 0  1]
 [-1  0]]

matrix_power(i, 3) :
 [[ 0 -1]
 [ 1  0]]

matrix_power(i, 0):
 [[1 0]
 [0 1]]

matrix_power(i, -3):
 [[ 0.  1.]
 [-1.  0.]]
```

© 2000-2024, MetaQuotes Ltd.

Dot

Produit scalaire de deux vecteurs.

```
double vector::Dot(
  const vector&  b       // deuxième vecteur
   );
```

Paramètres

b

[in] Vecteur.

Valeur de Retour

Un scalaire.

Note

Le produit scalaire de deux matrices est le produit matriciel matrix::MatMul().

Un algorithme simple pour le produit scalaire de vecteurs en MQL5 :

```
double VectorDot(const vector& vector_a, const vector& vector_b)
  {
   double dot=0;

   if(vector_a.Size()==vector_b.Size())
     {
      for(ulong i=0; i<vector_a.Size(); i++)
         dot+=vector_a[i]*vector_b[i];
     }

   return(dot);
  }
```

Exemple en MQL5 :

```
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
```

© 2000-2024, MetaQuotes Ltd.

Exemple en Python :

```
import numpy as np

a = [1, 0, 0, 1]
b = [4, 1, 2, 2]
print(np.dot(a, b))

>>> 6
```

© 2000-2024, MetaQuotes Ltd.

Kron

Renvoie le produit de Kronecker de deux matrices, matrice et vecteur, vecteur et matrice ou deux vecteurs.

```
matrix matrix::Kron(
  const matrix&  b      // deuxième matrice
   );

matrix matrix::Kron(
  const vector&  b      // vecteur
   );

matrix vector::Kron(
  const matrix&  b      // matrice
   );

matrix vector::Kron(
  const vector&  b      // deuxième vecteur
   );
```

Paramètres

b

[in] Deuxième matrice.

Valeur de Retour

Matrice.

Note

Le produit de Kronecker est également appelé multiplication matricielle par blocs.

Un algorithme simple pour le produit de Kronecker pour deux matrices en MQL5 :

```
matrix MatrixKronecker(const matrix& matrix_a,const matrix& matrix_b)
  {
   ulong  M=matrix_a.Rows();
   ulong  N=matrix_a.Cols();
   ulong  P=matrix_b.Rows();
   ulong  Q=matrix_b.Cols();
   matrix matrix_c(M*P,N*Q);

   for(ulong m=0; m<M; m++)
      for(ulong n=0; n<N; n++)
         for(ulong p=0; p<P; p++)
```

© 2000-2024, MetaQuotes Ltd.

```
            for(ulong q=0; q<Q; q++)
               matrix_c[m*P+p][n*Q+q]=matrix_a[m][n] * matrix_b[p][q];

   return(matrix_c);
  }
```

Exemple en MQL5 :

```
   matrix a={{1,2,3},{4,5,6}};
   matrix b=matrix::Identity(2,2);
   vector v={1,2};

   Print(a.Kron(b));
   Print(a.Kron(v));

  /*
   [[1,0,2,0,3,0]
    [0,1,0,2,0,3]
    [4,0,5,0,6,0]
    [0,4,0,5,0,6]]

   [[1,2,2,4,3,6]
    [4,8,5,10,6,12]]
  */
```

Exemple en Python :

```
import numpy as np

A = np.arange(1,7).reshape(2,3)
B = np.identity(2)
V = [1,2]
print(np.kron(A, B))
print("")
print(np.kron(A, V))

[[1. 0. 2. 0. 3. 0.]
 [0. 1. 0. 2. 0. 3.]
 [4. 0. 5. 0. 6. 0.]
 [0. 4. 0. 5. 0. 6.]]

[[ 1  2  2  4  3  6]
 [ 4  8  5 10  6 12]]
```

© 2000-2024, MetaQuotes Ltd.

Inner

Produit interne de deux matrices.

```
matrix matrix::Inner(
  const matrix&  b      // deuxième matrice
   );
```

Paramètres

b

[in] Matrice.

Valeur de Retour

Matrice.

Note

Le produit scalaire de deux vecteurs est le produit scalaire des deux vecteurs vector::Dot().

Un algorithme simple pour le produit interne de deux matrices en MQL5 :

```
bool MatrixInner(matrix& c, const matrix& a, const matrix& b)
  {
//--- le nombre de colonnes doit être égal
   if(a.Cols()!=b.Cols())
      return(false);
//--- la taille de la matrice résultante dépend du nombre de vecteurs dans chacune des
   ulong  rows=a.Rows();
   ulong  cols=b.Rows();
   matrix result(rows,cols);
//---
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
//---
   c=result;
   return(true);
  }
```

Exemple en MQL5 :

© 2000-2024, MetaQuotes Ltd.

```
   matrix a={{0,1,2},{3,4,5}};
   matrix b={{0,1,2},{3,4,5},{6,7,8}};
   matrix c=a.Inner(b);
   Print(c);
   matrix a1={{0,1,2}};
   matrix c1=a1.Inner(b);
   Print(c1);

  /*
  [[5,14,23]
  [14,50,86]]
  [[5,14,23]]
  */
```

Exemple en Python :

```
import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))

import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))
```

© 2000-2024, MetaQuotes Ltd.

Outer

Calcule le produit extérieur de deux matrices ou de deux vecteurs.

```
matrix matrix::Outer(
  const matrix&  b      // deuxième matrice
   );

matrix vector::Outer(
  const vector&  b      // deuxième vecteur
   );
```

Paramètres

b

[in] Matrice.

Valeur de Retour

Matrice.

Note

Le produit extérieur, comme le produit de Kronecker, est également une multiplication matricielle (et vectorielle) par blocs.

Un algorithme simple pour le produit extérieur de deux matrices en MQL5 :

```
matrix MatrixOuter(const matrix& matrix_a, const matrix& matrix_b)
  {
//--- la taille de la matrice résultante dépend des tailles de matrice
   ulong  rows=matrix_a.Rows()*matrix_a.Cols();
   ulong  cols=matrix_b.Rows()*matrix_b.Cols();
   matrix matrix_c(rows,cols);
   ulong  cols_a=matrix_a.Cols();
   ulong  cols_b=matrix_b.Cols();
//---
   for(ulong i=0; i<rows; i++)
     {
      ulong row_a=i/cols_a;
      ulong col_a=i%cols_a;
      for(ulong j=0; j<cols; j++)
        {
         ulong row_b=j/cols_b;
         ulong col_b=j%cols_b;
         matrix_c[i][j]=matrix_a[row_a][col_a] * matrix_b[row_b][col_b];
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//---
   return(matrix_c);
  }
```

Exemple en MQL5 :

```
   vector vector_a={0,1,2,3,4,5};
   vector vector_b={0,1,2,3,4,5,6};
   Print("vector_a.Outer\n",vector_a.Outer(vector_b));
   Print("vector_a.Kron\n",vector_a.Kron(vector_b));

   matrix matrix_a={{0,1,2},{3,4,5}};
   matrix matrix_b={{0,1,2},{3,4,5},{6,7,8}};
   Print("matrix_a.Outer\n",matrix_a.Outer(matrix_b));
   Print("matrix_a.Kron\n",matrix_a.Kron(matrix_b));

  /*
   vector_a.Outer
   [[0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6]
    [0,2,4,6,8,10,12]
    [0,3,6,9,12,15,18]
    [0,4,8,12,16,20,24]
    [0,5,10,15,20,25,30]]
   vector_a.Kron
   [[0,0,0,0,0,0,0,0,1,2,3,4,5,6,0,2,4,6,8,10,12,0,3,6,9,12,15,18,0,4,8,12,16,20,24,0,
   matrix_a.Outer
   [[0,0,0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6,7,8]
    [0,2,4,6,8,10,12,14,16]
    [0,3,6,9,12,15,18,21,24]
    [0,4,8,12,16,20,24,28,32]
    [0,5,10,15,20,25,30,35,40]]
   matrix_a.Kron
   [[0,0,0,0,1,2,0,2,4]
    [0,0,0,3,4,5,6,8,10]
    [0,0,0,6,7,8,12,14,16]
    [0,3,6,0,4,8,0,5,10]
    [9,12,15,12,16,20,15,20,25]
    [18,21,24,24,28,32,30,35,40]]
   */
```

Exemple en Python :

```
import numpy as np
```

© 2000-2024, MetaQuotes Ltd.

```
A = np.arange(6)
B = np.arange(7)
print("np.outer")
print(np.outer(A, B))
print("np.kron")
print(np.kron(A, B))

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
print("np.outer")
print(np.outer(A, B))
print("np.kron")
```

```
np.outer
[[ 0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6]
 [ 0  2  4  6  8 10 12]
 [ 0  3  6  9 12 15 18]
 [ 0  4  8 12 16 20 24]
 [ 0  5 10 15 20 25 30]]
np.kron
[ 0  0  0  0  0  0  0  0  1  2  3  4  5  6  0  2  4  6  8 10 12  0  3  6
  9 12 15 18  0  4  8 12 16 20 24  0  5 10 15 20 25 30]
np.outer
[[ 0  0  0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6  7  8]
 [ 0  2  4  6  8 10 12 14 16]
 [ 0  3  6  9 12 15 18 21 24]
 [ 0  4  8 12 16 20 24 28 32]
 [ 0  5 10 15 20 25 30 35 40]]
np.kron
[[ 0  0  0  0  1  2  0  2  4]
 [ 0  0  0  3  4  5  6  8 10]
 [ 0  0  0  6  7  8 12 14 16]
 [ 0  3  6  0  4  8  0  5 10]
 [ 9 12 15 12 16 20 15 20 25]
 [18 21 24 24 28 32 30 35 40]]
```

© 2000-2024, MetaQuotes Ltd.

CorrCoef

Calculer le coefficient de corrélation de Pearson (coefficient de corrélation linéaire).

```
matrix matrix::CorrCoef(
  const bool    rowvar=true  // vecteurs ligne ou colonne d'observations
);

scalar vector::CorrCoef(
  const vector&  b           // deuxième vecteur
);
```

Valeur de Retour

Coefficients de corrélation produit-moment de Pearson.

Note

Le coefficient de corrélation est dans la plage [-1, 1].

En raison de l'arrondi en virgule flottante, le tableau résultant peut ne pas être hermitien, les éléments diagonaux peuvent ne pas être 1 et les éléments peuvent ne pas satisfaire l'inégalité abs(a) <= 1. Les parties réelles et imaginaires sont coupées à l'intervalle [-1, 1] dans une tentative d'améliorer cette situation mais n'est pas d'une grande aide dans le cas complexe.

Un algorithme simple pour calculer le coefficient de corrélation de deux vecteurs en utilisant MQL5 :

```
double VectorCorrelation(const vector& vector_x,const vector& vector_y)
  {
   ulong n=vector_x.Size()<vector_y.Size() ? vector_x.Size() : vector_y.Size();
   if(n<=1)
      return(0);

   ulong  i;
   double xmean=0;
   double ymean=0;
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_x[i]))
         return(0);
      if(!MathIsValidNumber(vector_y[i]))
         return(0);
      xmean+=vector_x[i];
      ymean+=vector_y[i];
     }
   xmean/=(double)n;
```

© 2000-2024, MetaQuotes Ltd.

```
   ymean/=(double)n;

   double s=0;
   double xv=0;
   double yv=0;
   double t1=0;
   double t2=0;
//--- calcul
   s=0;
   for(i=0; i<n; i++)
     {
      t1=vector_x[i]-xmean;
      t2=vector_y[i]-ymean;
      xv+=t1*t1;
      yv+=t2*t2;
      s+=t1*t2;
     }
//--- vérification
   if(xv==0 || yv==0)
      return(0);
//--- retourne le résultat
   return(s/(MathSqrt(xv)*MathSqrt(yv)));
  }
```

Exemple en MQL5 :

```
   vectorf vector_a={1,2,3,4,5};
   vectorf vector_b={0,1,0.5,2,2.5};
   Print("corrélation des vecteurs ",vector_a.CorrCoef(vector_b));
//---
   matrixf matrix_a={{1,2,3,4,5},
                     {0,1,0.5,2,2.5}};
corrélation des lignes de la matrice   Print("matrix rows correlation\n",matrix_a.Corr
   matrixf matrix_a2=matrix_a.Transpose();
   Print("corrélation des colonnes transposées de la matrice\n",matrix_a2.CorrCoef(fal
   matrixf matrix_a3={{1.0f, 2.0f, 3.0f, 4.0f, 5.0f},
                      {0.0f, 1.0f, 0.5f, 2.0f, 2.5f},
                      {0.1f, 1.0f, 2.0f, 1.0f, 0.3f}};
   Print("corrélation des lignes\n",matrix_a3.CorrCoef());
   Print("corrélation des colonnes\n",matrix_a3.CorrCoef(false));

  /*
   corrélation des vecteurs 0.9149913787841797
   corrélation des lignes de la matrice
   [[1,0.91499138]
    [0.91499138,1]]
   corrélation  des colonnes transposées  de la matrice
   [[1,0.91499138]
```

© 2000-2024, MetaQuotes Ltd.

```
    [0.91499138,1]]
   corrélation des lignes
   [[1,0.91499138,0.08474271]
    [0.91499138,1,-0.17123166]
    [0.08474271,-0.17123166,1]]
   corrélation des colonnes
   [[1,0.99587059,0.85375023,0.91129309,0.83773589]
    [0.99587059,1,0.80295509,0.94491106,0.88385159]
    [0.85375023,0.80295509,1,0.56362146,0.43088508]
    [0.91129309,0.94491106,0.56362146,1,0.98827404]
    [0.83773589,0.88385159,0.43088508,0.98827404,1]]
   */
```

Exemple en Python :

```
import numpy as np
va=[1,2,3,4,5]
vb=[0,1,0.5,2,2.5]
print("corrélation des vecteurs")
print(np.corrcoef(va,vb))

ma=np.zeros((2,5))
ma[0,:]=va
ma[1,:]=vb
print("corrélation des lignes de la matrice")
print(np.corrcoef(ma))
print("corrélation des colonnes transposées de la matrice")
print(np.corrcoef(np.transpose(ma),rowvar=False))
print("")

ma1=[[1,2,3,4,5],[0,1,0.5,2,2.5],[0.1,1,0.2,1,0.3]]
print("corrélation des lignes\n",np.corrcoef(ma1))
print("corrélation des colonnes\n",np.corrcoef(ma1,rowvar=False))

corrélation des colonnes transposées de la matrice
[[1.         0.91499142]
 [0.91499142 1.        ]]

corrélation des lignes
 [[1.         0.91499142 0.1424941 ]
 [0.91499142 1.         0.39657517]
 [0.1424941  0.39657517 1.        ]]
corrélation des colonnes
 [[1.         0.99587059 0.98226063 0.91129318 0.83773586]
 [0.99587059 1.         0.99522839 0.94491118 0.88385151]
 [0.98226063 0.99522839 1.         0.97234063 0.92527551]
 [0.91129318 0.94491118 0.97234063 1.         0.98827406]
 [0.83773586 0.88385151 0.92527551 0.98827406 1.        ]]
```

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Cov

Calcule la matrice de covariance.

```
matrix matrix::Cov(
  const bool    rowvar=true  // vecteurs ligne ou colonne d'observations
);

matrix vector::Cov(
  const vector&  b           // deuxième vecteur
);
```

Paramètres

b

[in] Deuxième vecteur.

Note

Calcule la matrice de covariance.

Un algorithme simple pour calculer la matrice de covariance de deux vecteurs en utilisant MQL5 :

```
bool VectorCovariation(const vector& vector_a,const vector& vector_b,matrix& matrix_c)
  {
   int i,j;
   int m=2;
   int n=(int)(vector_a.Size()<vector_b.Size()?vector_a.Size():vector_b.Size());
//--- vérifications
   if(n<=1)
      return(false);
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_a[i]))
         return(false);
      if(!MathIsValidNumber(vector_b[i]))
         return(false);
     }
//---
   matrix matrix_x(2,n);
   matrix_x.Row(vector_a,0);
   matrix_x.Row(vector_b,1);
   vector t=vector::Zeros(m);
//--- calcul
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
         t[i]+=matrix_x[i][j]/double(n);
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
```

© 2000-2024, MetaQuotes Ltd.

```
         matrix_x[i][j]-=t[i];
//--- syrk C=alpha*A^H*A+beta*C (beta=0 et n'est pas considéré)
   matrix_c=matrix::Zeros(m,m);
   for(i=0; i<m; i++)
     {
      for(j=0; j<n; j++)
        {
         double v=matrix_x[i][j]/(n-1);
         for(int i_=i; i_<m; i_++)
            matrix_c[i][i_]+=v*matrix_x[i_][j];
        }
     }
//--- force la symétrie
   for(i=0; i<m-1; i++)
      for(j=i+1; j<m; j++)
         matrix_c[j][i]=matrix_c[i][j];
//---
   return(true);
  }
```

Exemple en MQL5 :

```
   matrix matrix_a={{3,-2.1},{1.1,-1},{0.12,4.3}};
   Print("covariation cols\n",matrix_a.Cov(false));
   Print("covariation rows\n",matrix_a.Cov());

   vector vector_a=matrix_a.Col(0);
   vector vector_b=matrix_a.Col(1);
   Print("covariation vectors\n",vector_a.Cov(vector_b));

  /*
   covariation cols
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
   covariation rows
   [[13.005,5.355,-10.659]
    [5.355,2.205,-4.389]
    [-10.659,-4.389,8.736199999999998]]
   covariation vectors
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
  */
```

Exemple en Python :

```
import numpy as np
matrix_a=np.array([[3,-2.1],[1.1,-1],[0.12,4.3]])
```

© 2000-2024, MetaQuotes Ltd.

```
matrix_c=np.cov(matrix_a,rowvar=False)
print("covariation cols\n",matrix_c)
matrix_c2=np.cov(matrix_a)
print("covariation rows\n",matrix_c2)

vector_a=matrix_a[:,0]
vector_b=matrix_a[:,1]
matrix_c3=np.cov(vector_a,vector_b)
print("covariation vectors\n",matrix_c3)

covariation cols
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
covariation rows
 [[ 13.005    5.355  -10.659 ]
 [  5.355    2.205   -4.389 ]
 [-10.659   -4.389    8.7362]]
covariation vectors
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
```

© 2000-2024, MetaQuotes Ltd.

Correlate

Calcule la corrélation croisée de deux vecteurs.

```
vector vector::Correlate(
  const vector&           v,         // vecteur
  ENUM_VECTOR_CONVOLVE    mode       // mode
   );
```

Paramètres

v

[in] Deuxième vecteur.

mode

[in] Le paramètre 'mode' détermine le mode de calcul de la convolution linéaire. Valeur de l'énumération ENUM_VECTOR_CONVOLVE.

Valeur de Retour

Intercorrélation de deux vecteurs.

Note

Le paramètre 'mode' détermine le mode de calcul de la convolution linéaire.

Un algorithme simple pour calculer le coefficient de corrélation de deux vecteurs en utilisant MQL5 :

```
vector VectorCrossCorrelationFull(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[n-i-1]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationSame(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n);
```

© 2000-2024, MetaQuotes Ltd.

```
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationValid(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
```

Exemple en MQL5 :

```
   vector a={1,2,3,4,5};
   vector b={0,1,0.5};

   Print("full\n",a.Correlate(b,VECTOR_CONVOLVE_FULL));
   Print("same\n",a.Correlate(b,VECTOR_CONVOLVE_SAME));
   Print("valid\n",a.Correlate(b,VECTOR_CONVOLVE_VALID));
   Print("full\n",b.Correlate(a,VECTOR_CONVOLVE_FULL));

  /*
```

© 2000-2024, MetaQuotes Ltd.

```
   full
   [0.5,2,3.5,5,6.5,5,0]
   same
   [2,3.5,5,6.5,5]
   valid
   [3.5,5,6.5]
   full
   [0,5,6.5,5,3.5,2,0.5]
  */
```

Exemple en Python :

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.correlate(a,b,'full'))
print("same\n",np.correlate(a,b,'same'));
print("valid\n",np.correlate(a,b,'valid'));
print("full\n",np.correlate(b,a,'full'))

full
 [0.5 2.  3.5 5.  6.5 5.  0. ]
same
 [2.  3.5 5.  6.5 5. ]
valid
 [3.5 5.  6.5]
full
 [0.  5.  6.5 5.  3.5 2.  0.5]
```

© 2000-2024, MetaQuotes Ltd.

Convolve

Renvoie la convolution discrète et linéaire de deux vecteurs

```
vector vector::Convolve(
  const vector&            v,         // vecteur
  ENUM_VECTOR_CONVOLVE     mode       // mode
   );
```

Paramètres

v

[out] Deuxième vecteur.

mode

[in] Le paramètre 'mode' détermine le mode de calcul de la convolution linéaire ENUM_VECTOR_CONVOLVE.

Valeur de Retour

Convolution discrète et linéaire de deux vecteurs.

Un algorithme simple pour calculer la convolution de deux vecteurs en MQL5 :

```
vector VectorConvolutionFull(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionFull(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[i]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionSame(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionSame(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
```

© 2000-2024, MetaQuotes Ltd.

```
   int    size=MathMax(m,n);
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionValid(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionValid(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
```

Exemple en MQL5 :

```
  vector a= {1, 2, 3, 4, 5};
  vector b= {0, 1, 0.5};

  Print("full\n", a.Convolve(b, VECTOR_CONVOLVE_FULL));
  Print("same\n", a.Convolve(b, VECTOR_CONVOLVE_SAME));
```

© 2000-2024, MetaQuotes Ltd.

```
  Print("valid\n", a.Convolve(b, VECTOR_CONVOLVE_VALID));

  /*
   full
   [0,1,2.5,4,5.5,7,2.5]
   same
   [1,2.5,4,5.5,7]
   valid
   [2.5,4,5.5]
  */
```

Exemple en Python :

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.convolve(a,b,'full'))
print("same\n",np.convolve(a,b,'same'));
print("valid\n",np.convolve(a,b,'valid'));


full
 [0.  1.  2.5 4.  5.5 7.  2.5]
same
 [1.  2.5 4.  5.5 7. ]
valid
 [2.5 4.  5.5]
```

© 2000-2024, MetaQuotes Ltd.

Transformations matricielles

La décomposition matricielle peut être utilisée dans les cas suivants :

- comme étape intermédiaire lors de la résolution de systèmes d'équations linéaires
- pour l'inversion de matrice
- lors du calcul des déterminants
- lors de la recherche des valeurs propres et des vecteurs propres d'une matrice
- lors du calcul des fonctions d'analyse des matrices
- lors de l'utilisation de la méthode des moindres carrés
- dans la solution numérique des équations différentielles

Différents types de décomposition matricielle sont utilisés en fonction du problème.

Fonction	Action
Cholesky	Calcule la décomposition de Cholesky
Eig	Calcule les valeurs propres et les vecteurs propres droits d'une matrice carrée
EigVals	Calcule les valeurs propres d'une matrice générale
LU	Factorisation LU d'une matrice comme étant le produit de la matrice triangulaire inférieure et de la matrice triangulaire supérieure
LUP	Factorisation LUP avec pivotement partiel, qui fait référence à la décomposition LU avec permutations de lignes uniquement : PA=LU
QR	Calcule la factorisation qr d'une matrice
SVD	Décomposition en Valeurs Singulières

© 2000-2024, MetaQuotes Ltd.

Cholesky

Calcule la décomposition de Cholesky.

```
bool matrix::Cholesky(
  matrix&  L      // matrice
   );
```

Paramètres

L key

[out] Matrice triangulaire inférieure.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Note

Retourne la décomposition de Cholesky, L * L.H, de la matrice carrée a, où L est triangulaire inférieur et .H est l'opérateur de transposition conjuguée (qui est la transposition ordinaire si a est à valeur réelle). a doit être Hermitien (symétrique si à valeur réelle) et défini positif. Aucune vérification n'est effectuée pour vérifier que a est Hermitien ou pas. De plus, seuls les éléments triangulaires inférieurs et diagonaux de a sont utilisés. Seul L est réellement retourné.

Exemple

```
matrix matrix_a= {{5.7998084, -2.1825367}, {-2.1825367, 9.85910595}};
matrix matrix_l;
Print("matrix_a\n", matrix_a);

matrix_a.Cholesky(matrix_l);
Print("matrix_l\n", matrix_l);
Print("check\n", matrix_l.MatMul(matrix_l.Transpose()));

/*
matrix_a
[[5.7998084,-2.1825367]
 [-2.1825367,9.85910595]]
matrix_l
[[2.408279136645086,0]
 [-0.9062640068544704,3.006291985133859]]
check
[[5.7998084,-2.1825367]
 [-2.1825367,9.85910595]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Eig

Calcule les valeurs propres et les vecteurs propres droits d'une matrice carrée.

```
bool matrix::Eig(
  matrix&  eigen_vectors,     // matrice de vecteurs propres
  vector&  eigen_values       // vecteur de valeurs propres
   );
```

Paramètres

eigen_vectors

[out] Matrice de vecteurs propres verticaux.

eigen_values

[out] Vecteur de valeurs propres.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Exemple

```
#property script_show_inputs
//--- Paramètres d'entrée
input int  InpSize1   =512;
input int  InpSize2   =256;
input int  InpSize3   =1024;
//+------------------------------------------------------------------+
//| Remplissage de la matrice carrée inversible de tests |
//+------------------------------------------------------------------+
template<typename T>
void MatrixFill(matrix<T> &matrix_a)
 {
  ulong size_m=matrix_a.Rows();
  ulong size_k=matrix_a.Cols();
  T     value=0.0;
//--- remplit la matrice
  for(ulong i=0; i<size_m; i++)
   {
    for(ulong j=0; j<size_k; j++)
     {
      if(i==j)
        matrix_a[i][j]=T(1.0+i);
      else
       {
        value+=1.0;
        matrix_a[i][j]=value;
       }
     }
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  }
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
int OnStart()
  {
   int errors=0;

   errors+=TestEigen<double>(InpSize1);
   errors+=TestEigen<double>(InpSize2);
   errors+=TestEigen<double>(InpSize3);

   errors+=TestEigen<float>(InpSize1);
   errors+=TestEigen<float>(InpSize2);
   errors+=TestEigen<float>(InpSize3);
//---
   Print("Test ", errors?"failed":"passed");
   return(errors);
  }

/*
Résultat

Eigen solver of double matrix 512 x 512  passed  vectors=489  time=4268.506 ms
Eigen solver of double matrix 256 x 256  passed  vectors=251  time=417.610 ms
Eigen solver of double matrix 1024 x 1024  passed  vectors=916  time=43708.280 ms
Eigen solver of float matrix 512 x 512  passed  vectors=1  time=2508.357 ms
Eigen solver of float matrix 256 x 256  passed  vectors=1  time=188.859 ms
Eigen solver of float matrix 1024 x 1024  passed  vectors=1  time=27209.666 ms
Test passed
*/

//+------------------------------------------------------------------+
//|  Test de la méthode Eig                                          |
//+------------------------------------------------------------------+
template<typename T>
int TestEigen(const int size_m)
  {
   int vectors=0;
   matrix<T> matrix_a(size_m, size_m);
   matrix<T> matrix_v(size_m, size_m);
   vector<T> vector_e(size_m);
//--- remplit la matrice carrée
   MatrixFill(matrix_a);
//--- les microsecondes seront mesurées
   ulong t1=GetMicrosecondCount();
//--- Solveur Eigen
   matrix_a.Eig(matrix_v, vector_e);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- mesure
  ulong t2=GetMicrosecondCount();
//--- check if A * v = lambda * v
  for(ulong n=0; n<vector_e.Size(); n++)
   {
    vector<T> eigen_vector=matrix_v.Col(n);
    vector<T> vector_c1   =eigen_vector*vector_e[n];
    vector<T> vector_c2   =matrix_a.MatMul(eigen_vector);

    //--- trop de divisions, la précision sera vérifiée à la 10ème décimale
    ulong errors=vector_c1.CompareByDigits(vector_c2, sizeof(T)==sizeof(double) ? 10 :
    if(int(errors)<size_m/10)
      vectors++;
   }
  double elapsed_time=double(t2-t1)/1000.0;
  printf("Eigen solver of %s matrix %d x %d  %s  vectors=%d  time=%.3f ms",
         typename(T), size_m, size_m, vectors>0?"passed":"failed", vectors, elapsed_ti
  return(vectors==0);
 }
```

© 2000-2024, MetaQuotes Ltd.

EigVals

Calcule les valeurs propres d'une matrice générale.

```
bool matrix::EigVals(
  vector&  eigen_values      // vecteur de valeurs propres
   );
```

Paramètres

eigen_values

[out] Vecteur de valeurs propres de droite.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Note

La seule différence entre EigVals et Eig est que EigVals ne calcule pas les vecteurs propres, alors qu'il ne calcule que les valeurs propres.

© 2000-2024, MetaQuotes Ltd.

LU

Factorisation LU d'une matrice comme étant le produit de la matrice triangulaire inférieure et de la matrice triangulaire supérieure.

```
bool matrix::LU(
  matrix&  L,      // matrice triangulaire inférieure
  matrix&  U       // matrice triangulaire supérieure
   );
```

Paramètres

L key

[out] Matrice triangulaire inférieure.

U

[out] Matrice triangulaire supérieure.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Exemple

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u;
//--- décomposition LU
   matrix_a.LU(matrix_l,matrix_u);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
//--- vérifie si A = L * U
   Print("vérification\n",matrix_l.MatMul(matrix_u));


   /*
   matrix_l
   [[1,0,0,0]
    [5,1,0,0]
    [8,0.875,1,0]
    [11,1.25,0.5904761904761905,1]]
   matrix_u
   [[1,2,3,4]
    [0,-8,-9,-13]
    [0,0,-13.125,-10.625]
    [0,0,0,-17.47619047619047]]
   vérification
   [[1,2,3,4]
```

© 2000-2024, MetaQuotes Ltd.

```
    [5,2,6,7]
    [8,9,3,10]
    [11,12,14,4]]
  */
```

© 2000-2024, MetaQuotes Ltd.

LUP

Factorisation LUP avec pivotement partiel, qui fait référence à la décomposition LU avec permutations de lignes uniquement : PA=LU.

```
bool  LUP(
  matrix&  L,      // matrice triangulaire inférieure
  matrix&  U,      // matrice triangulaire supérieure
  matrix&  P       // matrice de permutations
   );
```

Paramètres

L key

[out] Matrice triangulaire inférieure.

U

[out] Matrice triangulaire supérieure.

P

[out] Matrice de permutations

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Exemple

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u,matrix_p;
//--- décomposition LUP
   matrix_a.LUP(matrix_l,matrix_u,matrix_p);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
   Print("matrix_p\n",matrix_p);
//--- vérifie si P * A = L * U
   Print("P * A\n",matrix_p.MatMul(matrix_a));
   Print("L * U\n",matrix_l.MatMul(matrix_u));

   /*
   matrix_l
   [[1,0,0,0]
    [0.4545454545454545,1,0,0]
    [0.7272727272727273,-0.07894736842105282,1,0]
    [0.09090909090909091,-0.2631578947368421,-0.2262773722627738,1]]
   matrix_u
   [[11,12,14,4]
```

© 2000-2024, MetaQuotes Ltd.

```
 [0,-3.454545454545454,-0.3636363636363633,5.181818181818182]
 [0,0,-7.210526315789473,7.500000000000001]
 [0,0,0,6.697080291970803]]
matrix_p
[[0,0,0,1]
 [0,1,0,0]
 [0,0,1,0]
 [1,0,0,0]]
P * A
[[11,12,14,4]
 [5,2,6,7]
 [8,9,3,10]
 [1,2,3,4]]
L * U
[[11,12,14,4]
 [5,2,6,7]
 [8,9,3.000000000000001,10]
 [1,2,3,4]]
*/
```

© 2000-2024, MetaQuotes Ltd.

QR

Calcule la factorisation qr d'une matrice.

```
bool  QR(
  matrix&  Q,      // matrice avec des colonnes orthonormées
  matrix&  R       // matrice triangulaire supérieure
   );
```

Paramètres

Q

[out] Une matrice avec des colonnes orthonormées. Lorsque mode = 'complet' le résultat est une matrice orthogonale/unitaire selon que a est réel/complexe ou non. Le déterminant peut être +/- 1 dans ce cas. Si le nombre de dimensions dans le tableau d'entrée est supérieur à 2, une pile des matrices avec les propriétés ci-dessus est renvoyée.

R

[out] Matrice triangulaire supérieure.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Exemple

```
//---  A*x = b
  matrix A = {{0, 1}, {1, 1}, {1, 1}, {2, 1}};
  Print("A \n", A);
  vector b = {1, 2, 2, 3};
  Print("b \n", b);
//--- A = Q*R
  matrix q, r;
  A.QR(q, r);
  Print("q \n", q);
  Print("r \n", r);
  matrix qr=q.MatMul(r);
  Print("qr \n", qr);
  /*
  A
  [[0,1]
  [1,1]
  [1,1]
  [2,1]]
  b
  [1,2,2,3]
  q
  [[0.4082482904638631,-0.8164965809277259,-1.110223024625157e-16,-0.4082482904638631]
  [0.4625425214347352,-0.03745747856526496,0.7041241452319315,0.5374574785652647]
  [-0.5374574785652648,-0.03745747856526496,0.7041241452319316,-0.4625425214347352]
```

© 2000-2024, MetaQuotes Ltd.

```
[-0.5749149571305296,-0.5749149571305299,-0.09175170953613698,0.5749149571305296]]
r
[[-1.224744871391589,-0.2415816237971962]
[-1.22474487139159,-1.466326495188786]
[1.224744871391589,1.316496580927726]
[1.224744871391589,0.2415816237971961]]
qr
[[-1.110223024625157e-16,1]
[1,0.9999999999999999]
[1,1]
[2,1]]
*/
```

© 2000-2024, MetaQuotes Ltd.

SVD

Décomposition en Valeurs Singulières.

```
bool matrix::SVD(
  matrix&  U,                    // matrice unitaire
  matrix&  V,                    // matrice unitaire
  vector&  singular_values       // vecteur de valeurs singulières
   );
```

Paramètres

U

[out] Matrice unitaire d'ordre m, constituée de vecteurs singuliers gauches.

V

[out] Matrice unitaire d'ordre n, constituée de vecteurs singuliers droits.

singular_values

[out] Valeurs Singulières

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Exemple

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
//--- exécute la décomposition SVD
  matrix U, V;
  vector singular_values;
  b.SVD(U, V, singular_values);
  Print("U \n", U);
  Print("V \n", V);
  Print("singular_values = ", singular_values);

// vérification
//--- U * diagonale singulière * V = A
  matrix matrix_s;
  matrix_s.Diag(singular_values);
  Print("matrix_s \n", matrix_s);
  matrix matrix_vt=V.Transpose();
  Print("matrix_vt \n", matrix_vt);
  matrix matrix_usvt=(U.MatMul(matrix_s)).MatMul(matrix_vt);
  Print("matrix_usvt \n", matrix_usvt);
```

© 2000-2024, MetaQuotes Ltd.

```
  ulong errors=(int)b.Compare(matrix_usvt, 1e-9);
  double res=(errors==0);
  Print("errors=", errors);

//---- autre vérification
  matrix U_Ut=U.MatMul(U.Transpose());
  Print("U_Ut \n", U_Ut);
  Print("Ut_U \n", (U.Transpose()).MatMul(U));

  matrix vt_V=matrix_vt.MatMul(V);
  Print("vt_V \n", vt_V);
  Print("V_vt \n", V.MatMul(matrix_vt));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
  [[-4,-3,-2]
   [-1,0,1]
   [2,3,4]]
  U
  [[-0.7071067811865474,0.5773502691896254,0.408248290463863]
   [-6.827109697437648e-17,0.5773502691896253,-0.8164965809277256]
   [0.7071067811865472,0.5773502691896255,0.4082482904638627]]
  V
  [[0.5773502691896258,-0.7071067811865474,-0.408248290463863]
   [0.5773502691896258,1.779939029415334e-16,0.8164965809277258]
   [0.5773502691896256,0.7071067811865474,-0.408248290463863]]
  singular_values = [7.348469228349533,2.449489742783175,3.277709923350408e-17]

  matrix_s
  [[7.348469228349533,0,0]
   [0,2.449489742783175,0]
   [0,0,3.277709923350408e-17]]
  matrix_vt
  [[0.5773502691896258,0.5773502691896258,0.5773502691896256]
   [-0.7071067811865474,1.779939029415334e-16,0.7071067811865474]
   [-0.408248290463863,0.8164965809277258,-0.408248290463863]]
  matrix_usvt
  [[-3.999999999999997,-2.999999999999999,-2]
   [-0.9999999999999981,-5.977974170712231e-17,0.9999999999999974]
   [2,2.999999999999999,3.999999999999996]]
  erreurs=0

  U_Ut
  [[0.9999999999999993,-1.665334536937735e-16,-1.665334536937735e-16]
   [-1.665334536937735e-16,0.9999999999999987,-5.551115123125783e-17]
   [-1.665334536937735e-16,-5.551115123125783e-17,0.999999999999999]]
```

© 2000-2024, MetaQuotes Ltd.

```
 Ut_U
 [[0.9999999999999993,-5.551115123125783e-17,-1.110223024625157e-16]
  [-5.551115123125783e-17,0.9999999999999987,2.498001805406602e-16]
  [-1.110223024625157e-16,2.498001805406602e-16,0.999999999999999]]
 vt_V
 [[1,-5.551115123125783e-17,0]
  [-5.551115123125783e-17,0.9999999999999996,1.110223024625157e-16]
  [0,1.110223024625157e-16,0.9999999999999996]]
 V_vt
 [[0.9999999999999999,1.110223024625157e-16,1.942890293094024e-16]
  [1.110223024625157e-16,0.9999999999999998,1.665334536937735e-16]
  [1.942890293094024e-16,1.665334536937735e-16,0.9999999999999996]
 */
}
```

© 2000-2024, MetaQuotes Ltd.

Méthodes statistiques

Méthodes de calcul de statistiques descriptives de matrices et de vecteurs.

Fonction	Action
ArgMax	Renvoie l'indice de la valeur maximale
ArgMin	Renvoie l'indice de la valeur minimale
Max	Renvoie la valeur maximale dans une matrice/vecteur
Min	Renvoie la valeur minimale dans une matrice/vecteur
Ptp	Renvoie la plage de valeurs d'une matrice/vecteur ou de l'axe de matrice donné
Sum	Renvoie la somme des éléments de matrice/vecteur qui peuvent également être exécutés pour l'axe (les axes) donné(s)
Prod	Renvoie le produit des éléments matrice/vecteur qui peut également être effectué pour l'axe donné
CumSum	Renvoie la somme cumulée des éléments de matrice/vecteur, y compris ceux le long de l'axe donné
CumProd	Renvoie le produit cumulé des éléments de matrice/vecteur, y compris ceux le long de l'axe donné
Percentile	Renvoie le centile spécifié des valeurs des éléments matriciels/vectoriels ou des éléments le long de l'axe spécifié
Quantile	Renvoie le quantile spécifié des valeurs des éléments matriciels/vectoriels ou des éléments le long de l'axe spécifié
Median	Calcule la médiane des éléments matrice/vecteur
Mean	Calcule la moyenne arithmétique des valeurs des éléments
Average	Calcule la moyenne pondérée des valeurs de matrice/vecteur
Std	Renvoie l'écart type des valeurs des éléments de matrice/vecteur ou des éléments le long de l'axe donné
Var	Calcule la variance des valeurs des éléments de matrice/vecteur
LinearRegression	Calcule un vecteur/une matrice avec des valeurs de régression linéaire calculées

© 2000-2024, MetaQuotes Ltd.

ArgMax

Renvoie l'indice de la valeur maximale.

```
ulong vector::ArgMax();

ulong matrix::ArgMax();

vector matrix::ArgMax(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Indice de la valeur maximale.

Exemple

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.ArgMax(0);
   vector rows_max=matrix_a.ArgMax(1);
   ulong  matrix_max=matrix_a.ArgMax();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max index ",matrix_max,"  max value ",matrix_a.Flat(matrix_max));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[0,3,1]
   rows_max=[0,2,0,1]
   max index 5  max value 12.0
   */
```

© 2000-2024, MetaQuotes Ltd.

ArgMin

Renvoie l'indice de la valeur minimale.

```
ulong vector::ArgMin();

ulong matrix::ArgMin();

vector matrix::ArgMin(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Indice de la valeur minimale.

Exemple

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.ArgMin(0);
   vector rows_min=matrix_a.ArgMin(1);
   ulong  matrix_min=matrix_a.ArgMin();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min index ",matrix_min,"  min value ",matrix_a.Flat(matrix_min));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,0,0]
   rows_min=[2,0,2,0]
   min index 3  min value 1.0
   */
```

© 2000-2024, MetaQuotes Ltd.

Max

Renvoie la valeur maximale dans une matrice/un vecteur.

```
double vector::Max();

double matrix::Max();

vector matrix::Max(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Valeur maximale dans une matrice/un vecteur.

Exemple

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.Max(0);
   vector rows_max=matrix_a.Max(1);
   double matrix_max=matrix_a.Max();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max value ",matrix_max);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[10,11,12]
   rows_max=[10,12,6,11]
   max value 12.0
   */
```

© 2000-2024, MetaQuotes Ltd.

Min

Renvoie la valeur minimale dans une matrice/un vecteur.

```
double vector::Min();

double matrix::Min();

vector matrix::Min(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Valeur minimale dans une matrice/un vecteur.

Exemple

```
matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
Print("matrix_a\n",matrix_a);

vector cols_min=matrix_a.Min(0);
vector rows_min=matrix_a.Min(1);
double matrix_min=matrix_a.Min();

Print("cols_min=",cols_min);
Print("rows_min=",rows_min);
Print("min value ",matrix_min);

/*
matrix_a
[[10,3,2]
 [1,8,12]
 [6,5,4]
 [7,11,9]]
cols_min=[1,3,2]
rows_min=[2,1,4,7]
min value 1.0
*/
```

© 2000-2024, MetaQuotes Ltd.

Ptp

Renvoie la plage de valeurs d'une matrice/d'un vecteur ou de l'axe de matrice donné, équivalent à Max() - Min(). Ptp - Crête à crête.

```
double vector::Ptp();

double matrix::Ptp();

vector matrix::Ptp(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Vecteur avec plages de valeurs (maximum - minimum) .

Exemple

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_ptp=matrix_a.Ptp(0);
   vector rows_ptp=matrix_a.Ptp(1);
   double matrix_ptp=matrix_a.Ptp();

   Print("cols_ptp  ",cols_ptp);
   Print("rows_ptp  ",rows_ptp);
   Print("ptp value  ",matrix_ptp);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_ptp [9,8,10]
   rows_ptp [8,11,2,4]
   ptp value  11.0
   */
```

© 2000-2024, MetaQuotes Ltd.

Sum

Renvoie la somme des éléments de matrice/vecteur qui peuvent également être exécutés pour l'axe (les axes) donné(s).

```
double vector::Sum();

double matrix::Sum();

vector matrix::Sum(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Somme des éléments de matrice/vecteur qui peuvent également être exécutés pour l'axe (les axes) donné(s).

Exemple

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_sum=matrix_a.Sum(0);
   vector rows_sum=matrix_a.Sum(1);
   double matrix_sum=matrix_a.Sum();

   Print("cols_sum=",cols_sum);
   Print("rows_sum=",rows_sum);
   Print("sum value ",matrix_sum);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_sum=[24,27,27]
   rows_sum=[15,21,15,27]
   sum value 78.0
   */
```

© 2000-2024, MetaQuotes Ltd.

Prod

Renvoie le produit des éléments matrice/vecteur qui peut également être effectué pour l'axe donné.

```
double vector::Prod(
  const double  initial=1      // multiplicateur initial
   );

double matrix::Prod(
  const double  initial=1      // multiplicateur initial
   );

vector matrix::Prod(
  const int     axis,          // axe
  const double  initial=1      // multiplicateur initial
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

initial=1

[in] Multiplicateur initial.

Exemple

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_prod=matrix_a.Prod(0);
   vector rows_prod=matrix_a.Prod(1);
   double matrix_prod=matrix_a.Prod();

   Print("cols_prod=",cols_prod);
   cols_prod=matrix_a.Prod(0,0.1);
   Print("cols_prod=",cols_prod);
   Print("rows_prod=",rows_prod);
   Print("prod value ",matrix_prod);

  /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_prod=[420,1320,864]
   cols_prod=[42,132,86.40000000000001]
   rows_prod=[60,96,120,693]
```

© 2000-2024, MetaQuotes Ltd.

```
   prod value 479001600.0
  */
```

© 2000-2024, MetaQuotes Ltd.

CumSum

Renvoie la somme cumulée des éléments de matrice/vecteur, y compris ceux le long de l'axe donné.

```
vector vector::CumSum();

vector matrix::CumSum();

matrix matrix::CumSum(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Somme cumulée des éléments le long de l'axe donné.

Exemple

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumsum=matrix_a.CumSum(0);
   matrix rows_cumsum=matrix_a.CumSum(1);
   vector cumsum_values=matrix_a.CumSum();

   Print("cols_cumsum\n",cols_cumsum);
   Print("rows_cumsum\n",rows_cumsum);
   Print("cumsum values  ",cumsum_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumsum
   [[10,3,2]
    [11,11,14]
    [17,16,18]
    [24,27,27]]
   rows_cumsum
   [[10,13,15]
    [1,9,21]
    [6,11,15]
```

© 2000-2024, MetaQuotes Ltd.

```
 [7,18,27]]
cumsum values  [10,13,15,16,24,36,42,47,51,58,69,78]
*/
```

© 2000-2024, MetaQuotes Ltd.

CumProd

Renvoie le produit cumulé des éléments matrice/vecteur, y compris ceux le long de l'axe donné.

```
vector vector::CumProd();

vector matrix::CumProd();

matrix matrix::CumProd(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 - axe horizontal pour chaque colonne (c'est-à-dire sur les lignes), 1 - axe vertical pour chaque ligne (c'est-à-dire sur les colonnes)

Valeur de Retour

Produit cumulé des éléments le long de l'axe donné.

Exemple

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumprod=matrix_a.CumProd(0);
   matrix rows_cumprod=matrix_a.CumProd(1);
   vector cumprod_values=matrix_a.CumProd();

   Print("cols_cumprod\n",cols_cumprod);
   Print("rows_cumprod\n",rows_cumprod);
   Print("cumprod values  ",cumprod_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumprod
   [[10,3,2]
    [10,24,24]
    [60,120,96]
    [420,1320,864]]
   rows_cumprod
   [[10,30,60]
    [1,8,96]
```

© 2000-2024, MetaQuotes Ltd.

```
 [6,30,120]
 [7,77,693]]
cumprod values  [10,30,60,60,480,5760,34560,172800,691200,4838400,53222400,4790016(
*/
```

© 2000-2024, MetaQuotes Ltd.

Percentile

Renvoie le centile spécifié des valeurs des éléments de matrice/vecteur ou des éléments le long de l'axe spécifié.

```
double vector::Percentile(
  const int  percent      //
   );

double matrix::Percentile(
  const int  percent      //
   );

vector matrix::Percentile(
  const int  percent,     //
  const int  axis         // axe
   );
```

Paramètres

percent

[in] Centiles à calculer, qui doivent être compris entre 0 et 100 inclus.

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Centile : scalaire ou vecteur.

Note

Les valeurs valides du paramètre 'percent' sont comprises dans la plage [0, 100]. Un algorithme linéaire est utilisé pour calculer les centiles. Le calcul correct des percentiles nécessite que la séquence soit triée.

Exemple

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_percentile=matrix_a.Percentile(50,0);
   vectorf rows_percentile=matrix_a.Percentile(50,1);
   float matrix_percentile=matrix_a.Percentile(50);

   Print("cols_percentile ",cols_percentile);
   Print("rows_percentile ",rows_percentile);
   Print("percentile value  ",matrix_percentile);
```

© 2000-2024, MetaQuotes Ltd.

```
/*
matrix_a
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
cols_percentile [5.5,6.5,7.5]
rows_percentile [2,5,8,11]
percentile value  6.5
*/
```

© 2000-2024, MetaQuotes Ltd.

Quantile

Renvoie le quantile spécifié des valeurs des éléments de matrice/vecteur ou des éléments le long de l'axe spécifié.

```
double vector::Quantile(
  const double quantile      // quantile
   );

double matrix::Quantile(
  const double quantile      // quantile
   );

vector matrix::Quantile(
  const double quantile,     // quantile
  const int    axis          // axe
   );
```

Paramètres

quantile

[in] Quantile à calculer, qui doit être compris entre 0 et 1 inclus.

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Quantile : scalaire ou vecteur.

Note

Le paramètre 'quantile' prend des valeurs dans l'intervalle [0, 1]. Un algorithme linéaire est utilisé pour calculer les quantiles. Le calcul correct des quantiles nécessite que la séquence soit triée.

Exemple

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_quantile=matrix_a.Quantile(0.5,0);
   vectorf rows_quantile=matrix_a.Quantile(0.5,1);
   float matrix_quantile=matrix_a.Quantile(0.5);

   Print("cols_quantile ",cols_quantile);
   Print("rows_quantile ",rows_quantile);
   Print("quantile value  ",matrix_quantile);

   /*
   matrix_a
```

© 2000-2024, MetaQuotes Ltd.

```
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
cols_quantile [5.5,6.5,7.5]
rows_quantile [2,5,8,11]
quantile value  6.5
*/
```

© 2000-2024, MetaQuotes Ltd.

Median

Calcule la médiane des éléments de matrice/vecteur.

```
double vector::Median();

double matrix::Median();

vector matrix::Median(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Médiane : scalaire ou vecteur.

Note

La médiane est la valeur médiane qui sépare la moitié la plus élevée des éléments du tableau/vecteur de la moitié la plus basse des éléments. Identique à Quantile(0.5) et Centile(50). Le calcul correct des percentiles nécessite que la séquence soit triée.

Exemple

```
matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
Print("matrix_a\n",matrix_a);

vectorf cols_median=matrix_a.Median(0);
vectorf rows_median=matrix_a.Median(1);
float matrix_median=matrix_a.Median();

Print("cols_median ",cols_median);
Print("rows_median ",rows_median);
Print("median value  ",matrix_median);


/*
matrix_a
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
cols_median [5.5,6.5,7.5]
```

© 2000-2024, MetaQuotes Ltd.

```
rows_median [2,5,8,11]
median value  6.5
*/
```

© 2000-2024, MetaQuotes Ltd.

Mean

Calcule la moyenne arithmétique des valeurs des éléments.

```
double vector::Mean();

double matrix::Mean();

vector matrix::Mean(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Moyenne arithmétique des valeurs des éléments.

Exemple

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_mean=matrix_a.Mean(0);
   vectorf rows_mean=matrix_a.Mean(1);
   float matrix_mean=matrix_a.Mean();

   Print("cols_mean ",cols_mean);
   Print("rows_mean ",rows_mean);
   Print("mean value  ",matrix_mean);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_mean [6,6.75,6.75]
   rows_mean [5,7,5,9]
   mean value  6.5
   */
```

© 2000-2024, MetaQuotes Ltd.

Average

Calcule la moyenne pondérée des valeurs de matrice/vecteur.

```
double vector::Average(
  const vector& weigts      // vecteur de pondération
   );

double matrix::Average(
  const matrix& weigts      // matrice de pondération
);

vector matrix::Average(
  const matrix& weigts,     // matrice de pondération
  const int     axis        // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Moyenne arithmétique : scalaire ou vectorielle.

Note

La matrice/vecteur de pondération est associée à la matrice/vecteur principal.

Exemple

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   matrixf matrix_w=matrixf::Ones(4,3);
   Print("matrix_a\n",matrix_a);

   vectorf cols_average=matrix_a.Average(matrix_w,0);
   vectorf rows_average=matrix_a.Average(matrix_w,1);
   float matrix_average=matrix_a.Average(matrix_w);

   Print("cols_average ",cols_average);
   Print("rows_average ",rows_average);
   Print("average value  ",matrix_average);


   /*
   matrix_a
   [[10,3,2]
```

© 2000-2024, MetaQuotes Ltd.

```
 [1,8,12]
 [6,5,4]
 [7,11,9]]
cols_average [6,6.75,6.75]
rows_average [5,7,5,9]
average value  6.5
*/ value  6.5
```

© 2000-2024, MetaQuotes Ltd.

Std

Renvoie l'écart type des valeurs des éléments de matrice/vecteur ou des éléments le long de l'axe donné.

```
double vector::Std();

double matrix::Std();

vector matrix::Std(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Standard_deviation : scalaire ou vecteur.

Note

L'écart type est la racine carrée de la moyenne des écarts au carré par rapport à la moyenne, c'est à dire , std = sqrt(mean(x)), où x = abs(a - a.mean())**2.

L'écart quadratique moyen est généralement calculé sous la forme x.sum() / N, où N = len(x).

Exemple

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_std=matrix_a.Std(0);
   vectorf rows_std=matrix_a.Std(1);
   float matrix_std=matrix_a.Std();

   Print("cols_std ",cols_std);
   Print("rows_std ",rows_std);
   Print("std value  ",matrix_std);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
```

© 2000-2024, MetaQuotes Ltd.

```
cols_std [3.2403703,3.0310888,3.9607449]
rows_std [3.5590262,4.5460606,0.81649661,1.6329932]
std value  3.452052593231201
*/
```

© 2000-2024, MetaQuotes Ltd.

Var

Calcule la variance des valeurs des éléments de matrice/vecteur.

```
double vector::Var();

double matrix::Var();

vector matrix::Var(
  const int  axis      // axe
   );
```

Paramètres

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

Variance : scalaire ou vectorielle.

Note

La variance est la moyenne des écarts au carré par rapport à la moyenne, c'est-à-dire var = moyenne(x), où x = abs(a - a.mean())**2.

La moyenne est généralement calculée comme x.sum() / N, où N = len(x).

Exemple

```
matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
Print("matrix_a\n",matrix_a);

vectorf cols_var=matrix_a.Var(0);
vectorf rows_var=matrix_a.Var(1);
float matrix_var=matrix_a.Var();

Print("cols_var ",cols_var);
Print("rows_var ",rows_var);
Print("var value  ",matrix_var);


/*
matrix_a
[[10,3,2]
 [1,8,12]
 [6,5,4]
 [7,11,9]]
cols_var [10.5,9.1875,15.6875]
```

© 2000-2024, MetaQuotes Ltd.

```
rows_var [12.666667,20.666666,0.66666669,2.6666667]
var value  11.916666984558105
*/
```

© 2000-2024, MetaQuotes Ltd.

LinearRegression

Calcule un vecteur/une matrice avec des valeurs de régression linéaire calculées.

```
vector vector::LinearRegression();

matrix matrix::LinearRegression(
  ENUM_MATRIX_AXIS  axis=AXIS_NONE      // axe le long duquel la régression est clacul
   );
```

Paramètres

axis

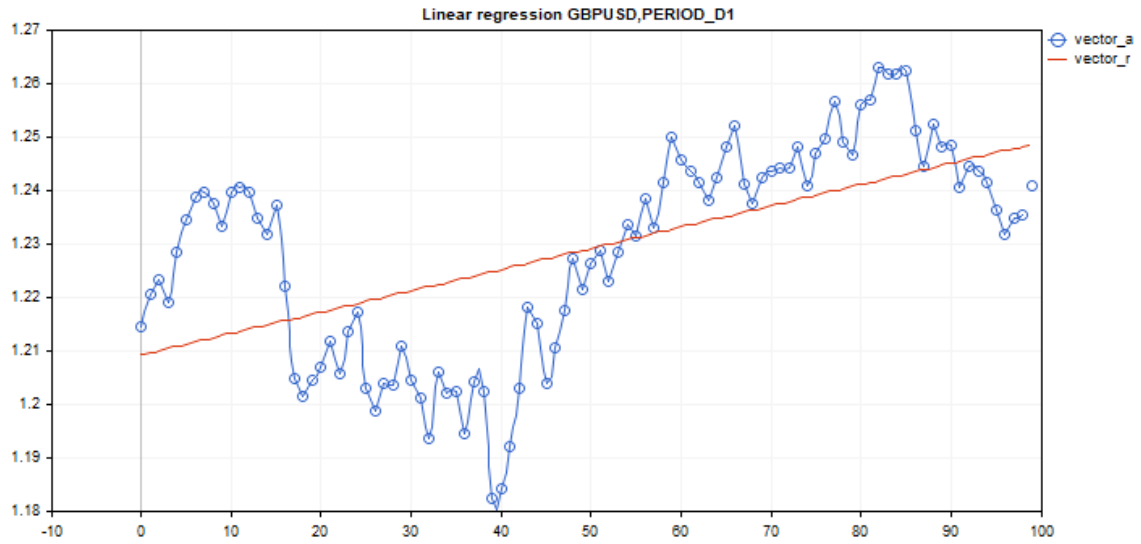
[in] Spécifie l'axe long duquel la régression est calculée. Valeur de l'énumération ENUM_MATRIX_AXIS (AXIS_HORZ — axe horizontal, AXIS_VERT — axe vertical).

Valeur de Retour

Vecteur ou matrice avec valeurs de régression linéaire calculées.

Note

La régression linéaire est calculée à l'aide de l'équation de régression standard : y (x) = a * x + b, où a est la pente de la ligne, et b son décalage sur l'axe Y.



Exemple :

```
#include <Graphics\Graphic.mqh>

#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Fonction de démarrage du programme                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector vector_a;
   vector_a.CopyRates(_Symbol,_Period,COPY_RATES_CLOSE,1,100);
   vector vector_r=vector_a.LinearRegression();

//--- désactive l'affichage du graphique
   ChartSetInteger(0,CHART_SHOW,false);

//--- tableaux pour dessiner le graphique
   double x[];
   double y1[];
   double y2[];
   ArrayResize(x,uint(vector_a.Size()));
   ArrayResize(y1,uint(vector_a.Size()));
   ArrayResize(y2,uint(vector_a.Size()));
   for(ulong i=0; i<vector_a.Size(); i++)
     {
      x[i]=(double)i;
      y1[i]=vector_a[i];
      y2[i]=vector_r[i];
     }

//--- titre du graphique
   string title="Régression linéaire "+_Symbol+","+EnumToString(_Period);

   long   chart=0;
   string name="LinearRegression";

//--- crée le graphique
   CGraphic graphic;
   graphic.Create(chart,name,0,0,0,GRAPH_WIDTH,GRAPH_HEIGHT);
   graphic.BackgroundMain(title);
   graphic.BackgroundMainSize(12);

//--- graphique de la fonction d'activation
   CCurve *curvef=graphic.CurveAdd(x,y1,CURVE_POINTS_AND_LINES);
   curvef.Name("vector_a");
   curvef.LinesWidth(2);
   curvef.LinesSmooth(true);
   curvef.LinesSmoothTension(1);
   curvef.LinesSmoothStep(10);

//--- dérivées de la fonction d'activation
   CCurve *curved=graphic.CurveAdd(x,y2,CURVE_LINES);
   curved.Name("vector_r");
   curved.LinesWidth(2);
```

© 2000-2024, MetaQuotes Ltd.

```
   curved.LinesSmooth(true);
   curved.LinesSmoothTension(1);
   curved.LinesSmoothStep(10);
   graphic.CurvePlotAll();
   graphic.Update();

//--- boucle infinie pour identifier les touches du clavier utilisées
   while(!IsStopped())
     {
      //--- appuyez sur Echap pour terminer le programme
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
         break;
      //--- appuyez sur Page Suivante pour sauvegarder une image du graphique
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0)
        {
         string file_names[];
         if(FileSelectDialog("Sauvegarder l'Image",NULL,"Tous les fichiers (*.*)|*.*",
            continue;
         ChartScreenShot(0,file_names[0],GRAPH_WIDTH,GRAPH_HEIGHT);
        }
      Sleep(10);
     }

//--- nettoyage
   graphic.Destroy();
   ObjectDelete(chart,name);
   ChartSetInteger(0,CHART_SHOW,true);
  }
```

ENUM_MATRIX_AXIS

Enumération permettant de spécifier l'axe dans toutes les fonctions statistiques pour les matrices.

ID	Description
AXIS_NONE	L'axe n'est pas spécifié. Le calcul est effectué sur tous les éléments de la matrice, comme s'il s'agissait d'un vecteur (voir la méthode Flat).
AXIS_HORZ	Axe horizontal
AXIS_VERT	Axe vertical

© 2000-2024, MetaQuotes Ltd.

Méthodes des fonctionnalités

Ces méthodes permettent des fonctionnalités sur les matrice, telles que :

- le nombre de lignes
- le nombre de colonnes
- la norme
- le nombre conditionnel
- le déterminant
- le rang de la matrice
- la trace
- le spectre

Fonction	Action
Rows	Retourne le nombre de lignes d'une matrice
Cols	Retourne le nombre de colonnes d'une matrice
Size	Retourne la taille d'un vecteur
Norm	Retourne la norme d'une matrice ou d'un vecteur
Cond	Calcule le conditionnement d'une matrice
Det	Calcule le déterminant d'une matrice carrée inversible
SLogDet	Calcule le signe et le logarithme du déterminant d'une matrice
Rank	Renvoie le rang de la matrice à l'aide de la méthode Gaussienne
Trace	Retourne la somme des diagonales d'une matrice
Spectrum	Calcule le spectre d'une matrice comme l'ensemble de ses valeurs propres du produit AT*A

© 2000-2024, MetaQuotes Ltd.

Rows

Retourne le nombre de lignes d'une matrice.

```
ulong matrix::Rows()
```

Valeur de Retour

Un entier

Exemple

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Rows()=", m.Rows());
Print("m.Cols()=", m.Cols());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Rows()=3
 m.Cols()=4
*/
```

© 2000-2024, MetaQuotes Ltd.

Cols

Retourne le nombre de colonnes d'une matrice.

```
ulong matrix::Cols()
```

Valeur de Retour

Un entier

Exemple

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Cols()=", m.Cols());
Print("m.Rows()=", m.Rows());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Cols()=4
 m.Rows()=3
*/
```

© 2000-2024, MetaQuotes Ltd.

Size

Rentourne la taille du vecteur.

```
ulong vector::Size()
```

Valeur de Retour

Un entier

Exemple

```
matrix m={{1,2,3,4,5,6,7,8,9,10,11,12}};
m.Reshape(3,4);
Print("matrix m\n",m);
vector v=m.Row(1);
Print("v.Size()=",v.Size());
Print("v=",v);


/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 v.Size()=4
 v=[5,6,7,8]
*/
```

© 2000-2024, MetaQuotes Ltd.

Norm

Retourne la norme de la matrice ou du vecteur.

```
double vector::Norm(
  const ENUM_VECTOR_NORM  norm,      // norme vectorielle
  const int               norm_p=2  // nombre de p-norm en cas de VECTOR_NORM_P
   );

double matrix::Norm(
  const ENUM_MATRIX_NORM  norm       // norme matricielle
   );
```

Paramètres

la norme

[in] Ordre de la norme

Valeur de Retour

Norme de la matrice ou du vecteur.

Note

- VECTOR_NORM_INF est la valeur absolue maximale parmi les éléments vectoriels.
- VECTOR_NORM_MINUS_INF est la valeur absolue minimale d'un vecteur.
- VECTOR_NORM_P est la norme P du vecteur. Si norm_p=0, alors c'est le nombre d'éléments vectoriels non nuls. norm_p=1 est la somme des valeurs absolues des éléments du vecteur. norm_p=2 est la racine carrée de la somme des carrés des valeurs des éléments vectoriels. La norme P peut être négative.
- MATRIX_NORM_FROBENIUS est la racine carrée de la somme des carrés des valeurs des éléments de la matrice. La norme de Frobenius et la norme vectorielle P2 sont cohérentes.
- MATRIX_NORM_SPECTRAL est la valeur maximale du spectre matriciel.
- MATRIX_NORM_NUCLEAR est la somme des valeurs singulières de la matrice.
- MATRIX_NORM_INF est le vecteur p1-norm maximal parmi les vecteurs verticaux de la matrice. La norme inf de la matrice et la norme inf du vecteur sont cohérentes.
- MATRIX_NORM_MINUS_INF est le vecteur p1-norm minimum parmi les vecteurs verticaux de la matrice.
- MATRIX_NORM_P1 est le vecteur p1-norm maximal parmi les vecteurs matriciels horizontaux.
- MATRIX_NORM_MINUS_P1 est le vecteur minimal p1-norm parmi les vecteurs matriciels horizontaux.
- MATRIX_NORM_P2 est la valeur singulière la plus élevée de la matrice.
- MATRIX_NORM_MINUS_P2 est la plus petite valeur singulière d'une matrice.

Un algorithme simple pour calculer la norme P d'un vecteur en MQL5 :

```
double VectorNormP(const vector& v,int norm_value)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   ulong  i;
   double norm=0.0;
//---
   switch(norm_value)
     {
      case 0 :
         for(i=0; i<v.Size(); i++)
            if(v[i]!=0)
               norm+=1.0;
         break;
      case 1 :
         for(i=0; i<v.Size(); i++)
            norm+=MathAbs(v[i]);
         break;
      case 2 :
         for(i=0; i<v.Size(); i++)
            norm+=v[i]*v[i];
         norm=MathSqrt(norm);
         break;
      default :
         for(i=0; i<v.Size(); i++)
            norm+=MathPow(MathAbs(v[i]),norm_value);
         norm=MathPow(norm,1.0/norm_value);
     }
//---
   return(norm);
  }
```

Exemple en MQL5 :

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_FROBENIUS)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_INF)", b.Norm(MATRIX_NORM_INF));
  Print("b.Norm(MATRIX_NORM_MINUS_INF)", b.Norm(MATRIX_NORM_MINUS_INF));
  Print("b.Norm(MATRIX_NORM_P1)=)", b.Norm(MATRIX_NORM_P1));
  Print("b.Norm(MATRIX_NORM_MINUS_P1)=", b.Norm(MATRIX_NORM_MINUS_P1));
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_P2));
  Print("b.Norm(MATRIX_NORM_MINUS_P2)=", b.Norm(MATRIX_NORM_MINUS_P2));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
```

© 2000-2024, MetaQuotes Ltd.

```
  [[-4,-3,-2]
  [-1,0,1]
  [2,3,4]]
  b.Norm(MATRIX_NORM_P2)=7.745966692414834
  b.Norm(MATRIX_NORM_FROBENIUS)=7.745966692414834
  b.Norm(MATRIX_NORM_INF)9.0
  b.Norm(MATRIX_NORM_MINUS_INF)2.0
  b.Norm(MATRIX_NORM_P1)=)7.0
  b.Norm(MATRIX_NORM_MINUS_P1)=6.0
  b.Norm(MATRIX_NORM_P2)=7.348469228349533
  b.Norm(MATRIX_NORM_MINUS_P2)=1.857033188519056e-16
  */
```

Exemple en Python :

```
import numpy as np
from numpy import linalg as LA
a = np.arange(9) - 4
print("a \n",a)
b = a.reshape((3, 3))
print("b \n",b)
print("LA.norm(b)=",LA.norm(b))
print("LA.norm(b, 'fro')=",LA.norm(b, 'fro'))
print("LA.norm(b, np.inf)=",LA.norm(b, np.inf))
print("LA.norm(b, -np.inf)=",LA.norm(b, -np.inf))
print("LA.norm(b, 1)=",LA.norm(b, 1))
print("LA.norm(b, -1)=",LA.norm(b, -1))
print("LA.norm(b, 2)=",LA.norm(b, 2))
print("LA.norm(b, -2)=",LA.norm(b, -2))

a
 [-4 -3 -2 -1  0  1  2  3  4]
b
 [[-4 -3 -2]
 [-1  0  1]
 [ 2  3  4]]
LA.norm(b)= 7.745966692414834
LA.norm(b, 'fro')= 7.745966692414834
LA.norm(b, np.inf)= 9.0
LA.norm(b, -np.inf)= 2.0
LA.norm(b, 1)= 7.0
LA.norm(b, -1)= 6.0
LA.norm(b, 2)= 7.3484692283495345
LA.norm(b, -2)= 1.857033188519056e-16
```

© 2000-2024, MetaQuotes Ltd.

Cond

Calcule le conditionnement d'une matrice.

```
double matrix::Cond(
  const ENUM_MATRIX_NORM  norm      // norme matricielle
   );
```

Paramètres

la norme

[in] Ordre de la norme depuis l'énumération ENUM_MATRIX_NORM

Valeur de Retour

Le conditionnement de la matrice. Peut être infini.

Note

Le conditionnement de x est défini comme la norme de x multipliée par la norme de l'inverse de x [1]. La norme peut être la norme L2 habituelle (racine de la somme des carrés) ou l'une des nombreuses autres normes matricielles.

Le conditionnement est la valeur K égale au produit des normes de la matrice A et de son inverse. Les matrices avec un conditionnement élevé sont dites mal conditionnées. Celles qui ont un conditionnement faible sont appelés bien conditionnées. La matrice inverse est obtenue par pseudo-inversion, afin de ne pas être limité par la condition des carrés et de non-singularité de la matrice.

Une exception est le conditionnement du spectre.

Un algorithme simple pour calculer le conditionnement du spectre en MQL5 :

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

  return(norm);
 }
```

Exemple en MQL5 :

```
matrix a= {{1, 0, -1}, {0, 1, 0}, { 1, 0, 1}};
Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
Print("a.Cond(MATRIX_NORM_FROBENIUS)=", a.Cond(MATRIX_NORM_FROBENIUS));
Print("a.Cond(MATRIX_NORM_INF)=", a.Cond(MATRIX_NORM_INF));
Print("a.Cond(MATRIX_NORM_MINUS_INF)=", a.Cond(MATRIX_NORM_MINUS_INF));
Print("a.Cond(MATRIX_NORM_P1)=)", a.Cond(MATRIX_NORM_P1));
Print("a.Cond(MATRIX_NORM_MINUS_P1)=", a.Cond(MATRIX_NORM_MINUS_P1));
Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
Print("a.Cond(MATRIX_NORM_MINUS_P2)=", a.Cond(MATRIX_NORM_MINUS_P2));

/*
matrix a
[[1,0,-1]
[0,1,0]
[1,0,1]]
a.Cond(MATRIX_NORM_P2)=1.414213562373095
a.Cond(MATRIX_NORM_FROBENIUS)=3.162277660168379
a.Cond(MATRIX_NORM_INF)=2.0
a.Cond(MATRIX_NORM_MINUS_INF)=0.9999999999999997
a.Cond(MATRIX_NORM_P1)=)2.0
a.Cond(MATRIX_NORM_MINUS_P1)=0.9999999999999998
a.Cond(MATRIX_NORM_P2)=1.414213562373095
a.Cond(MATRIX_NORM_MINUS_P2)=0.7071067811865472
*/
```

Exemple en Python :

```
import numpy as np
from numpy import linalg as LA
a = np.array([[1, 0, -1], [0, 1, 0], [1, 0, 1]])
print("a \n",a)
print("LA.cond(a)=",LA.cond(a))
print("LA.cond(a, 'fro')=",LA.cond(a, 'fro'))
print("LA.cond(a, np.inf)=",LA.cond(a, np.inf))
print("LA.cond(a, -np.inf)=",LA.cond(a, -np.inf))
print("LA.cond(a, 1)=",LA.cond(a, 1))
```

© 2000-2024, MetaQuotes Ltd.

```
print("LA.cond(a, -1)=",LA.cond(a, -1))
print("LA.cond(a, 2)=",LA.cond(a, 2))
print("LA.cond(a, -2)=",LA.cond(a, -2))

a
 [[ 1  0 -1]
 [ 0  1  0]
 [ 1  0  1]]
LA.cond(a)= 1.4142135623730951
LA.cond(a, 'fro')= 3.1622776601683795
LA.cond(a, np.inf)= 2.0
LA.cond(a, -np.inf)= 1.0
LA.cond(a, 1)= 2.0
LA.cond(a, -1)= 1.0
LA.cond(a, 2)= 1.4142135623730951
LA.cond(a, -2)= 0.7071067811865475
```

© 2000-2024, MetaQuotes Ltd.

Det

Calculer le déterminant d'une matrice carrée inversible.

```
double matrix::Det()
```

Valeur de Retour

Déterminant de la matrice.

Note

Les déterminants matriciels de 2e et 3e ordres sont calculés selon la règle de Sarrus. d2=a11*a22-a12*a21; d3=a11*a22*a33+a12*a23*a31+a13*a21*a32-a13*a22*a31-a11*a23*a32-a12*a21*a33

Le déterminant est calculé par la méthode Gaussienne en réduisant la matrice à une forme triangulaire supérieure. Le déterminant d'une matrice triangulaire supérieure est égal au produit des principaux éléments diagonaux.

Si au moins une ligne ou une colonne de la matrice est nulle, le déterminant est nul.

Si deux ou plusieurs lignes ou colonnes de la matrice sont linéairement dépendantes, son déterminant est égal à zéro.

Le déterminant d'une matrice est égal au produit de ses valeurs propres.

Exemple en MQL5 :

```
matrix m={{1,2},{3,4}};
double det=m.Det();
Print("matrix m\n",m);
Print("det(m)=",det);
/*
matrix m
[[1,2]
 [3,4]]
det(m)=-2.0
*/
```

Exemple en Python :

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
print('a \n',a)
print('nnp.linalg.det(a) \n',np.linalg.det(a))

a
 [[1 2]
```

© 2000-2024, MetaQuotes Ltd.

```
 [3 4]]

np.linalg.det(a)
 -2.0000000000000004
```

© 2000-2024, MetaQuotes Ltd.

SLogDet

Calculer le signe et le logarithme d'un déterminant de matrice.

```
double matrix::SLogDet(
  int&  sign      // sign
   );
```

Paramètres

sign

[out] Le signe du déterminant. Si le signe est pair, le déterminant est positif.

Valeur de Retour

Un nombre représentant le signe du déterminant.

Note

Le déterminant est calculé par la méthode Gaussienne en réduisant la matrice à une forme triangulaire supérieure. Le déterminant d'une matrice triangulaire supérieure est égal au produit des principaux éléments diagonaux. Le logarithme d'un produit est égal à la somme des logarithmes. Ainsi, en cas de débordement lors du calcul du déterminant, vous pouvez utiliser la méthode SLogDet.

Si le signe est pair, le déterminant est positif.

Exemple

```
a = np.array([[1, 2], [3, 4]])  (sign, logdet) = np.linalg.slogdet(a)  (sign, logdet)
```

© 2000-2024, MetaQuotes Ltd.

Rank

Retourne le rang de la matrice à l'aide de la méthode Gaussienne.

```
int  Rank()
```

Valeur de Retour

Rang de la matrice.

Note

Le rang d'un système de lignes (ou colonnes) d'une matrice A qui a m lignes et n colonnes est le nombre maximal de lignes (ou colonnes) linéairement indépendantes. Plusieurs lignes (colonnes) sont dites linéairement indépendantes si aucune d'entre elles ne peut être exprimée linéairement par rapport aux autres. Le rang du système de lignes est toujours égal au rang du système de colonnes. Cette valeur s'appelle le rang de la matrice.

Exemple en MQL5 :

```
matrix a=matrix::Eye(4, 4);;
Print("matrix a \n", a);
Print("a.Rank()=", a.Rank());

matrix I=matrix::Eye(4, 4);
I[3, 3] = 0.;    // rank deficient matrix
Print("I \n", I);
Print("I.Rank()=", I.Rank());

matrix b=matrix::Ones(1, 4);
Print("b \n", b);
Print("b.Rank()=", b.Rank());;// 1 dimension - rank 1 unless all 0

matrix  zeros=matrix::Zeros(4, 1);
Print("zeros \n", zeros);
Print("zeros.Rank()=", zeros.Rank());

/*
matrix a
[[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[0,0,0,1]]
a.Rank()=4

I
[[1,0,0,0]
[0,1,0,0]
```

© 2000-2024, MetaQuotes Ltd.

```
  [0,0,1,0]
  [0,0,0,0]]
  I.Rank()=3

  b
  [[1,1,1,1]]
  b.Rank()=1

  zeros
  [[0]
  [0]
  [0]
  [0]]
  zeros.Rank()=0
  */
```

Exemple en Python :

```
import numpy as np
from numpy.linalg import matrix_rank
a=(np.eye(4)) # Full rank matrix
print("a \n", a)
print("matrix_rank(a)=",matrix_rank(a))
I=np.eye(4)
I[-1,-1] = 0. # rank deficient matrix
print("I \n",I)
print("matrix_rank(I)=",matrix_rank(I))

b=np.ones((4,))
print("b \n",b)
print("matrix_rank(b)=",matrix_rank(b)) # 1 dimension - rank 1 unless all 0

zeros=np.zeros((4,))
print("zeroes \n",zeros)
print("matrix_rank(zeros)=",matrix_rank(zeros))

a
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
matrix_rank(a)= 4

I
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]]
```

© 2000-2024, MetaQuotes Ltd.

```
matrix_rank(I)= 3

b
 [1. 1. 1. 1.]
matrix_rank(b)= 1

zeroes
 [0. 0. 0. 0.]
matrix_rank(zeros)= 0
```

© 2000-2024, MetaQuotes Ltd.

Trace

Retourne la somme le long des diagonales de la matrice.

```
double matrix::Trace()
```

Valeur de Retour

La somme le long des diagonales.

Note

La trace d'une matrice est égale à la somme de ses valeurs propres.

Exemple en MQL5 :

```
matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
a.Reshape(3, 3);
Print("matrix a \n", a);
Print("a.Trace() \n", a.Trace());

/*
matrix a
[[0,1,2]
[3,4,5]
[6,7,8]]
a.Trace()
12.0
*/
```

Exemple en Python :

```
a = np.arange(9).reshape((3,3))
print('a \n',a)
print('np.trace(a) \n',np.trace(a))

a
 [[0 1 2]
 [3 4 5]
 [6 7 8]]

np.trace(a)
 12
```

© 2000-2024, MetaQuotes Ltd.

Spectrum

Calcule le spectre d'une matrice comme l'ensemble de ses valeurs propres à partir du produit AT*A.

```
vector matrix::Spectrum()
```

Valeur de Retour

Le spectre d'une matrice sous la forme d'un vecteur des valeurs propres de la matrice.

Exemple

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
        }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

© 2000-2024, MetaQuotes Ltd.

Méthodes matricielles pour résoudre des systèmes d'équations linéaires

Méthodes de résolution de systèmes d'équations linéaires et de calcul de la matrice inverse.

Fonction	Action
Solve	Résout une équation matricielle linéaire ou un système d'équations algébriques linéaires
LstSq	Renvoie la solution des moindres carrés des équations algébriques linéaires (pour les matrices non carrées ou dégénérées)
Inv	Calcule l'inverse (multiplicatif) d'une matrice carrée non dégénérée par la méthode de Jordan-Gauss
PInv	Calcule la pseudo-inverse d'une matrice par la méthode de Moore-Penrose

© 2000-2024, MetaQuotes Ltd.

Solve

Résout une équation matricielle linéaire ou un système d'équations algébriques linéaires.

```
vector matrix::Solve(
  const vector  b      // valeurs de l'ordonnée ou de la "variable dépendante"
   );
```

Paramètres

b

[in] Valeurs ordonnées ou 'variables dépendantes'. (Vecteur de termes libres).

Valeur de Retour

Vecteur avec solution du système a * x = b.

Note

Si au moins une ligne ou une colonne de la matrice est nulle, le système n'a pas de solution.

Si deux lignes ou colonnes de matrice ou plus sont linéairement dépendantes, le système n'a pas de solution.

Exemple

```
//--- solution SLAE
   vector_x=matrix_a.Solve(vector_b);
//--- vérifie si a * x = b
   result_vector=matrix_a.MatMul(vector_x);
   errors=vector_b.Compare(result_vector,1e-12);
```

© 2000-2024, MetaQuotes Ltd.

LstSq

Renvoie la solution des moindres carrés des équations algébriques linéaires (pour les matrices non carrées ou dégénérées).

```
vector matrix::LstSq(
  const vector  b      // valeurs de l'ordonnée ou de la "variable dépendante"
   );
```

Paramètres

b

[in] Valeurs ordonnées ou 'variables dépendantes'. (Vecteur de termes libres)

Valeur de Retour

Vecteur avec solution du système a * x = b. Ceci n'est vrai que pour les systèmes qui ont une solution exacte.

Exemple

```
   matrix a={{3, 2},
             {4,-5},
             {3, 3}};
   vector b={7,40,3};
//---
   vector x=a.LstSq(b);
//--- vérification, doit être [5, -4]
   Print("x=", x);
//--- vérifiecation, doit être [7, 40, 3]
   vector b1=a.MatMul(x);
   Print("b1=",b1);

/*
  x=[5.000000000000002,-4]
  b1=[7.000000000000005,40.00000000000001,3.000000000000005]
*/
```

© 2000-2024, MetaQuotes Ltd.

Inv

Calcule l'inverse multiplicatif d'une matrice carrée inversible par la méthode de Jordan-Gauss.

```
matrix matrix::Inv()
```

Valeur de Retour

Inverse multiplicatif de la matrice.

Note

Le produit de la matrice originale et de la matrice inverse est la matrice identité.

Si au moins une ligne ou colonne de la matrice est nulle, la matrice inverse ne peut pas être obtenue.

Si deux lignes ou colonnes de matrice ou plus sont linéairement dépendantes, la matrice inverse ne peut pas être obtenue.

Exemple

```
int TestInverse(const int size_m)
  {
   int    i,j,errors=0;
   matrix matrix_a(size_m,size_m);
//--- remplit la matrice carrée
   MatrixTestFirst(matrix_a);
//--- les microsecondes seront mesurées
   ulong t1=GetMicrosecondCount();
//--- calcule la matrice inverse
   matrix inverse=matrix_a.Inv();
//--- mesure
   ulong t2=GetMicrosecondCount();
//--- vérifie l'exactitude
   matrix identity=matrix_a.MatMul(inverse);
//---
   for(i=0; i<size_m; i++)
     {
      for(j=0; j<size_m; j++)
        {
         double value;
         //--- des 1 doivent être le long de la diagonale
         if(i==j)
            value=1.0;
         else
            value=0.0;
         if(MathClassify(identity[i][j])>FP_ZERO)
            errors++;
```

© 2000-2024, MetaQuotes Ltd.

```
         else
           {
            if(identity[i][j]!=value)
              {
               double diff=MathAbs(identity[i][j]-value);
               //--- trop de multiplications et de divisions, donc réduisez la précisi
               if(diff>1e-9)
                  errors++;
              }
           }
        }
     }
//---
   double elapsed_time=double(t2-t1)/1000.0;
   printf("Inversion de la matrice %d x %d  %s  erreurs=%d  time=%.3f ms",size_m,size_
   return(errors);
  }
```

© 2000-2024, MetaQuotes Ltd.

PInv

Calcule la pseudo-inverse d'une matrice par la méthode de Moore-Penrose.

```
matrix matrix::PInv()
```

Valeur de Retour

La pseudo-inverse de matrice.

Exemple

```
int TestPseudoInverse(const int size_m, const int size_k)
  {
   matrix matrix_a(size_m,size_k);
   matrix matrix_inverted(size_k,size_m);
   matrix matrix_temp;
   matrix matrix_a2;
//--- remplit la matrice
   MatrixTestFirst(matrix_a);
//--- inverse
   matrix_inverted=matrix_a.PInv();
//--- vérifie l'exactitude
   int errors=0;
//--- A * A+ * A = A   (A+ est un pseudo-inverse de A)
   matrix_temp=matrix_a.MatMul(matrix_inverted);
   matrix_a2=matrix_temp.MatMul(matrix_a);
   errors=(int)matrix_a.CompareByDigits(matrix_a2,10);

   printf("PseudoInversion %s matrix_size  %d x %d  erreurs=%d",errors==0?"passed":"fa
//---
   return(errors);
  }
```

© 2000-2024, MetaQuotes Ltd.

Machine learning

Ces méthodes sont utilisées en apprentissage automatique.

La fonction d'activation du réseau neuronal détermine la valeur de sortie d'un neurone en fonction de la somme pondérée des entrées. La sélection de la fonction d'activation a un impact important sur les performances du réseau neuronal. Différentes parties de modèle (couches) peuvent utiliser différentes fonctions d'activation.

En plus de toutes les fonctions d'activation connues, MQL5 propose également leurs dérivés. Les dérivées de fonction permettent une mise à jour efficace des paramètres du modèle en fonction de l'erreur reçue lors de l'apprentissage.

Un réseau de neurones vise à trouver un algorithme qui minimise l'erreur d'apprentissage, pour lequel la fonction de perte est utilisée. La valeur de la fonction de perte indique de combien la valeur prédite par le modèle s'écarte de la valeur réelle. Différentes fonctions de perte sont utilisées en fonction du problème. Par exemple, l'Erreur Quadratique Moyenne (Mean Squared Error, MSE) est utilisée pour les problèmes de régression, et l'Entropie Croisée Binaire (Binary Cross-Entropy, BCE) est utilisé à des fins de classification binaire.

Fonction	Action
Activation	Calcule les valeurs de la fonction d'activation et les écrit dans le vecteur/matrice passé
Derivative	Calcule les valeurs dérivées de la fonction d'activation et les écrit dans le vecteur/matrice passé
Loss	Calcule les valeurs de la fonction de perte et les écrit dans le vecteur/matrice passé
Loss	Calcule un vecteur ou une matrice de gradients de fonction de perte
RegressionMetric	Calculer la métrique de régression en tant qu'erreur d'écart par rapport à la ligne de régression construite sur le tableau de données spécifié
ConfusionMatrix	Calcule la matrice de confusion. La méthode est appliquée au vecteur de valeurs prédites
ConfusionMatrixMultilabel	Calcule la matrice de confusion pour chaque étiquette. La méthode est appliquée au vecteur de valeurs prédites
ClassificationMetric	Calcule la métrique de classification pour évaluer la qualité des données prédites par rapport aux données vraies. La méthode est appliquée au vecteur de valeurs prédites
ClassificationScore	Calcule la métrique de classification pour évaluer la qualité des données prédites par rapport aux données réelles

Exemple

Cet exemple illustre la formation d'un modèle à l'aide d'opérations matricielles. Le modèle est entraîné pour la fonction (a + b + c)^2 / (a^2 + b^2 + c^2). Nous saisissons la matrice de données initiale, dans

© 2000-2024, MetaQuotes Ltd.

laquelle a, b et c sont contenus dans des colonnes différentes. Le résultat de la fonction est obtenu en sortie du modèle.

```
matrix weights1, weights2, weights3;                // matrices de poids
matrix output1, output2, result;                    // matrices des sorties de la couch
input int layer1 = 200;                             // la taille de la première couche
input int layer2 = 200;                             // la taille de la deuxième couche
input int Epochs = 20000;                           // le nombre de passes d'entraîneme
input double lr = 3e-6;                             // taux d'apprentissage
input ENUM_ACTIVATION_FUNCTION ac_func = AF_SWISH; // fonction d'activation
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int train = 1000;    // taille de l'échantillon d'entraînement
   int test = 10;       // taille de l'échantillon de test
   matrix m_data, m_target;
//--- génére un échantillon d'apprentissage
   if(!CreateData(m_data, m_target, train))
      return;
//--- entraîne le modèle
   if(!Train(m_data, m_target, Epochs))
      return;
//--- génére un échantillon de test
   if(!CreateData(m_data, m_target, test))
      return;
//--- teste le modèle
   Test(m_data, m_target);
  }
//+------------------------------------------------------------------+
//| Méthode de génération d'échantillons                             |
//+------------------------------------------------------------------+
bool CreateData(matrix &data, matrix &target, const int count)
  {
//--- initialise les données initiales et les matrices de résultats
   if(!data.Init(count, 3) || !target.Init(count, 1))
      return false;
//--- remplit la matrice de données initiale avec des valeurs aléatoires
   data.Random(-10, 10);
//--- calcule les valeurs cibles pour l'échantillon d'apprentissage
   vector X1 = MathPow(data.Col(0) + data.Col(1) + data.Col(1), 2);
   vector X2 = MathPow(data.Col(0), 2) + MathPow(data.Col(1), 2) + MathPow(data.Col(2)
   if(!target.Col(X1 / X2, 0))
      return false;
//--- retourne le résultat
   return true;
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Méthode d'entraînement du modèle                                 |
//+------------------------------------------------------------------+
bool Train(matrix &data, matrix &target, const int epochs = 10000)
  {
//--- crée le modèle
   if(!CreateNet())
      return false;
//--- entraîne le modèle
   for(int ep = 0; ep < epochs; ep++)
     {
      //--- passe anticipée
      if(!FeedForward(data))
         return false;
      PrintFormat("Passe %d, loss %.5f", ep, result.Loss(target, LOSS_MSE));
      //--- rétropropagation et mise à jour de la matrice de poids
      if(!Backprop(data, target))
         return false;
     }
//--- retourne le résultat
   return true;
  }
//+------------------------------------------------------------------+
//| Méthode de création du modèle                                    |
//+------------------------------------------------------------------+
bool CreateNet()
  {
//--- initialise les matrices de poids
   if(!weights1.Init(4, layer1) || !weights2.Init(layer1 + 1, layer2) || !weights3.Ini
      return false;
//--- remplit les matrices de poids avec des valeurs aléatoires
   weights1.Random(-0.1, 0.1);
   weights2.Random(-0.1, 0.1);
   weights3.Random(-0.1, 0.1);
//--- retourne le résultat
   return true;
  }
//+------------------------------------------------------------------+
//| Méthode d'anticipation                                           |
//+------------------------------------------------------------------+
bool FeedForward(matrix &data)
  {
//--- vérifie la taille initiale des données
   if(data.Cols() != weights1.Rows() - 1)
      return false;
//--- calcule la première couche neuronale
   matrix temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
```

© 2000-2024, MetaQuotes Ltd.

```
      return false;
   output1 = temp.MatMul(weights1);
//--- calcule la fonction d'activation
   if(!output1.Activation(temp, ac_func))
      return false;
//--- calcule la deuxième couche neuronale
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   output2 = temp.MatMul(weights2);
//--- calcule la fonction d'activation
   if(!output2.Activation(temp, ac_func))
      return false;
//--- calcule la troisième couche neurale
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   result = temp.MatMul(weights3);
//--- retourne le résultat
   return true;
  }
//+------------------------------------------------------------------+
//| Méthode de rétropropagation                                      |
//+------------------------------------------------------------------+
bool Backprop(matrix &data, matrix &target)
  {
//--- vérifie la taille de la matrice des valeurs cibles
   if(target.Rows() != result.Rows() ||
      target.Cols() != result.Cols())
      return false;
//--- détermine l'écart des valeurs calculées par rapport à la cible
   matrix loss = (target - result) * 2;
//--- propage le gradient à la couche précédente
   matrix gradient = loss.MatMul(weights3.Transpose());
//--- met à jour la matrice de poids de la dernière couche
   matrix temp;
   if(!output2.Activation(temp, ac_func))
      return false;
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   weights3 = weights3 + temp.Transpose().MatMul(loss) * lr;
//--- ajuste le gradient d'erreur par la dérivée de la fonction d'activation
   if(!output2.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- propage le gradient à une couche inférieure
```

© 2000-2024, MetaQuotes Ltd.

```
   gradient = loss.MatMul(weights2.Transpose());
//--- met à jour la matrice de poids de la deuxième couche cachée
   if(!output1.Activation(temp, ac_func))
      return false;
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   weights2 = weights2 + temp.Transpose().MatMul(loss) * lr;
//--- ajuste le gradient d'erreur par la dérivée de la fonction d'activation
   if(!output1.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- met à jour la matrice de poids de la première couche cachée
   temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
   weights1 = weights1 + temp.Transpose().MatMul(loss) * lr;
//--- retourne le résultat
   return true;
  }
//+------------------------------------------------------------------+
//| Méthode de test du modèle                                        |
//+------------------------------------------------------------------+
bool Test(matrix &data, matrix &target)
  {
//--- propagation avant (feed forward) des données de test
   if(!FeedForward(data))
      return false;
//--- journalise les résultats du calcul du modèle et les valeurs vraies
   PrintFormat("Test loss %.5f", result.Loss(target, LOSS_MSE));
   ulong total = data.Rows();
   for(ulong i = 0; i < total; i++)
      PrintFormat("(%.2f + %.2f + %.2f)^2 / (%.2f^2 + %.2f^2 + %.2f^2) =  Net %.2f, Ta
                  data[i, 0], data[i, 1], data[i, 2], result[i, 0], target[i, 0]);
//--- retourne le résultat
   return true;
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

Activation

Calcule les valeurs de la fonction d'activation et les écrit dans le vecteur (ou la matrice) passé.

```
bool vector::Activation(
  vector&                    vect_out,      // vecteur pour obtenir les valeurs
  ENUM_ACTIVATION_FUNCTION   activation,    // fonction d'activation
   ...                                      // paramètres supplémentaires
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // matrice pour obtenir les valeurs
  ENUM_ACTIVATION_FUNCTION   activation     // fonction d'activation
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // matrice pour obtenir les valeurs
  ENUM_ACTIVATION_FUNCTION   activation,    // fonction d'activation
  ENUM_MATRIX_AXIS           axis,          // axe
   ...                                      // paramètres supplémentaires
   );
```

Paramètres

vect_out/matrix_out

[out] Vecteur ou matrice pour récupérer les valeurs calculées de la fonction d'activation.

activation

[in] Fonction d'activation de l'énumération ENUM_ACTIVATION_FUNCTION.

axis

[in] Valeur de l'énumération ENUM_MATRIX_AXIS (AXIS_HORZ - axe horizontal, AXIS_VERT - axe vertical).

...

[in] Paramètres supplémentaires requis pour certaines fonctions d'activation. Si aucun paramètre n'est spécifié, les valeurs par défaut sont utilisées.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Paramètres Supplémentaires

Certaines fonctions d'activation acceptent des paramètres supplémentaires. Si aucun paramètre n'est spécifié, les valeurs par défaut sont utilisées.

© 2000-2024, MetaQuotes Ltd.

```
AF_ELU  (Exponential Linear Unit)
  double alpha=1.0

Fonction d'activation : if(x>=0) f(x) = x
                   else f(x) = alpha * (exp(x)-1)


AF_LINEAR
  double alpha=1.0
  double beta=0.0

Fonction d'activation : f(x) = alpha*x + beta


AF_LRELU   (Leaky REctified Linear Unit)
  double alpha=0.3

Fonction d'activation : if(x>=0) f(x)=x
                   else f(x) = alpha*x


AF_RELU  (REctified Linear Unit)
  double alpha=0.0
  double max_value=0.0
  double treshold=0.0

Fonction d'activation : if(alpha==0) f(x) = max(x,0)
                   else if(x>max_value) f(x) = x
                   else f(x) = alpha*(x - treshold)


AF_SWISH
  double beta=1.0

Fonction d'activation : f(x) = x / (1+exp(-x*beta))


AF_TRELU   (Thresholded REctified Linear Unit)
  double theta=1.0

Fonction d'activation : if(x>theta) f(x) = x
                   else f(x) = 0


AF_PRELU   (Parametric REctified Linear Unit)
  double alpha[] - tableau appris de coefficients
```

© 2000-2024, MetaQuotes Ltd.

```
Fonction d'activation : if(x[i]>=0) f(x)[i] = x[i]
                        else f(x)[i] = alpha[i] * x[i]
```

Note

Dans les réseaux de neurones artificiels, la fonction d'activation d'un neurone détermine le signal de sortie, défini par un signal d'entrée ou un ensemble de signaux d'entrée. La sélection de la fonction d'activation a un impact important sur les performances du réseau de neurones. Différentes parties du modèle (couches) peuvent utiliser différentes fonctions d'activation.

Exemples d'utilisation de paramètres supplémentaires :

```
vector x={0.1, 0.4, 0.9, 2.0, -5.0, 0.0, -0.1};
vector y;

x.Activation(y,AF_ELU);
Print(y);
x.Activation(y,AF_ELU,2.0);
Print(y);

Print("");
x.Activation(y,AF_LINEAR);
Print(y);
x.Activation(y,AF_LINEAR,2.0);
Print(y);
x.Activation(y,AF_LINEAR,2.0,5.0);
Print(y);

Print("");
x.Activation(y,AF_LRELU);
Print(y);
x.Activation(y,AF_LRELU,1.0);
Print(y);
x.Activation(y,AF_LRELU,0.1);
Print(y);

Print("");
x.Activation(y,AF_RELU);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5,1.0);
Print(y);

Print("");
x.Activation(y,AF_SWISH);
Print(y);
```

© 2000-2024, MetaQuotes Ltd.

```
   x.Activation(y,AF_SWISH,2.0);
   Print(y);

   Print("");
   x.Activation(y,AF_TRELU);
   Print(y);
   x.Activation(y,AF_TRELU,0.3);
   Print(y);

   Print("");
   vector a=vector::Full(x.Size(),2.0);
   x.Activation(y,AF_PRELU,a);
   Print(y);

/*  Résultats
   [0.1,0.4,0.9,2,-0.993262053000915,0,-0.095162581964040]
   [0.1,0.4,0.9,2,-1.986524106001829,0,-0.190325163928081]

   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.2,0.8,1.8,4,-10,0,-0.2]
   [5.2,5.8,6.8,9,-5,5,4.8]

   [0.1,0.4,0.9,2,-1.5,0,-0.03]
   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.1,0.4,0.9,2,-0.5,0,-0.01]

   [0.1,0.4,0.9,2,0,0,0]
   [0.2,0.8,0.9,2,-10,0,-0.2]
   [-1.8,-1.2,0.9,2,-12,-2,-2.2]

   [0.052497918747894,0.239475064044981,0.6398545523625035,1.761594155955765,-0.033464
   [0.054983399731247,0.275989792451045,0.7723340415895611,1.964027580075817,-0.000226

   [0,0,0,2,0,0,0]
   [0,0.4,0.9,2,0,0,0]

   [0.1,0.4,0.9,2,-10,0,-0.2]
*/
```

© 2000-2024, MetaQuotes Ltd.

Derivative

Calcule les valeurs de la dérivée de la fonction d'activation et les écrits dans le vecteur ou la matrice passé en paramètre.

```
bool vector::Derivative(
  vector&                    vect_out,      // vecteur pour obtenir les valeurs
  ENUM_ACTIVATION_FUNCTION   activation,    // fonction d'activation
   ...                                      // paramètres supplémentaires
   );


bool matrix::Derivative(
  matrix&                    matrix_out,    // matrice pour obtenir les valeurs
  ENUM_ACTIVATION_FUNCTION   activation,    // fonction d'activation
   );


bool matrix::Derivative(
  matrix&                    matrix_out,    // matrice pour obtenir les valeurs
  ENUM_ACTIVATION_FUNCTION   activation,    // fonction d'activation
  ENUM_MATRIX_AXIS           axis,          // axe
   ...                                      // paramètres supplémentaires
   );
```

Paramètres

vect_out/matrix_out

[out] Vecteur ou matrice pour obtenir les valeurs calculées de la dérivée de la fonction d'activation.

activation

[in] Fonction d'activation de l'énumération ENUM_ACTIVATION_FUNCTION.

axis

[in] Valeur de l'énumération ENUM_MATRIX_AXIS (AXIS_HORZ - axe horizontal, AXIS_VERT - axe vertical).

...

[in] Les paramètres supplémentaires sont les mêmes que ceux des fonctions d'activation. Seules certaines fonctions d'activation acceptent des paramètres supplémentaires. Si aucun paramètre n'est spécifié, les valeurs par défaut sont utilisées.

Valeur de Retour

Renvoie true en cas de succès, false sinon.

Note

© 2000-2024, MetaQuotes Ltd.

Les dérivées de fonction permettent une mise à jour efficace des paramètres du modèle en fonction de l'erreur reçue lors de l'apprentissage lors de la rétropropagation de l'erreur.

© 2000-2024, MetaQuotes Ltd.

Loss

Calculer la valeur de la fonction de perte.

```
double vector::Loss(
  const vector&       vect_true,      // vecteur de valeurs vraies
  ENUM_LOSS_FUNCTION  loss,           // fonction de perte
   ...                                // paramètre supplémentaire
   );


double matrix::Loss(
  const matrix&       matrix_true,    // matrice de valeurs vraies
  ENUM_LOSS_FUNCTION  loss,           // fonction de perte
   );


double matrix::Loss(
  const matrix&       matrix_true,    // matrice de valeurs vraies
  ENUM_LOSS_FUNCTION  loss,           // fonction de perte
  ENUM_MATRIX_AXIS    axis,           // axe
   ...                                // paramètre supplémentaire
   );
```

Paramètres

vect_true/matrix_true

[in] Vecteur ou matrice de valeurs vraies.

loss

[in] Fonction de perte de l'énumération ENUM_LOSS_FUNCTION.

axis

[in] Valeur de l'énumération ENUM_MATRIX_AXIS (AXIS_HORZ - axe horizontal, AXIS_VERT - axe vertical).

...

[in] Le paramètre supplémentaire "delta" ne peut être utilisé que par la fonction de perte de Hubert (LOSS_HUBER)

Valeur de Retour

Valeur de type double.

Comment le paramètre 'delta' est utilisé dans la fonction de perte de Hubert (LOSS_HUBER)

```
   double delta = 1.0;
   double error = fabs(y - x);
   if(error<delta)
      loss = 0.5 * error^2;
```

© 2000-2024, MetaQuotes Ltd.

```
   else
      loss = 0.5 * delta^2 + delta * (error - delta);
```

Note

Un réseau de neurones vise à trouver les algorithmes qui minimisent l'erreur sur l'échantillon d'apprentissage, pour lequel la fonction de perte est utilisée.

La valeur de la fonction de perte indique de combien la valeur prédite par le modèle s'écarte de la valeur réelle.

Différentes fonctions de perte sont utilisées en fonction du problème. Par exemple, l'erreur quadratique moyenne (MSE) est utilisée pour les problèmes de régression, et l'entropie croisée binaire (BCE) est utilisée à des fins de classification binaire.

Exemple d'appel de la fonction de perte Hubert :

```
   vector y_true = {0.0, 1.0, 0.0, 0.0};
   vector y_pred = {0.6, 0.4, 0.4, 0.6};
   double loss=y_pred.Loss(y_true,LOSS_HUBER);
   Print(loss);
   double loss2=y_pred.Loss(y_true,LOSS_HUBER,0.5);
   Print(loss2);

/* Résultat
   0.155
   0.15125
*/
```

© 2000-2024, MetaQuotes Ltd.

LossGradient

Calcule un vecteur ou une matrice de gradients de fonction de perte.

```
vector vector::LossGradient(
  const vector&       vect_true,      // vecteur de valeurs vraies
  ENUM_LOSS_FUNCTION  loss,           // type de la fonction de perte
   ...                                // paramètre supplémentaire
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,    // matrice de valeurs vraies
  ENUM_LOSS_FUNCTION  loss,           // fonction de perte
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,    // matrice de valeurs vraies
  ENUM_LOSS_FUNCTION  loss,           // fonction de perte
  ENUM_MATRIX_AXIS    axis,           // axe
   ...                                // paramètre supplémentaire
   );
```

Paramètres

vect_true/matrix_true

[in] Vecteur ou matrice de valeurs vraies.

loss

[in] Fonction de perte de l'énumération ENUM_LOSS_FUNCTION.

axis

[in] Valeur de l'énumération ENUM_MATRIX_AXIS (AXIS_HORZ - axe horizontal, AXIS_VERT - axe vertical).

...

[in] Le paramètre supplémentaire "delta" ne peut être utilisé que par la fonction de perte de Hubert (LOSS_HUBER)

Valeur de Retour

Vecteur ou matrice des valeurs de gradient de la fonction de perte. Le gradient est la dérivée partielle par rapport à dx (x est la valeur prédite) de la fonction de perte en un point donné.

Note

Les gradients sont utilisés dans les réseaux de neurones pour ajuster les poids de la matrice de poids lors de la rétropropagation, lors de la formation du modèle.

© 2000-2024, MetaQuotes Ltd.

Un réseau de neurones vise à trouver les algorithmes qui minimisent l'erreur sur l'échantillon d'apprentissage, pour lequel la fonction de perte est utilisée.

Différentes fonctions de perte sont utilisées en fonction du problème. Par exemple, l'erreur quadratique moyenne (MSE) est utilisée pour les problèmes de régression, et l'entropie croisée binaire (BCE) est utilisée à des fins de classification binaire.

Exemple de calcul des gradients de la fonction de perte

```
   matrixf y_true={{ 1, 2, 3, 4 },
                   { 5, 6, 7, 8 },
                   { 9,10,11,12 }};
   matrixf y_pred={{ 1, 2, 3, 4 },
                   {11,10, 9, 8 },
                   { 5, 6, 7,12 }};
   matrixf loss_gradient =y_pred.LossGradient(y_true,LOSS_MAE);
   matrixf loss_gradienth=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_HORZ);
   matrixf loss_gradientv=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_VERT);
   Print("loss gradients\n",loss_gradient);
   Print("loss gradients on horizontal axis\n",loss_gradienth);
   Print("loss gradients on vertical axis\n",loss_gradientv);

/* Résultat
   loss gradients
   [[0,0,0,0]
    [0.083333336,0.083333336,0.083333336,0]
    [-0.083333336,-0.083333336,-0.083333336,0]]
   loss gradients on horizontal axis
   [[0,0,0,0]
    [0.33333334,0.33333334,0.33333334,0]
    [-0.33333334,-0.33333334,-0.33333334,0]]
   loss gradients on vertical axis
   [[0,0,0,0]
    [0.25,0.25,0.25,0]
    [-0.25,-0.25,-0.25,0]]
*/
```

© 2000-2024, MetaQuotes Ltd.

RegressionMetric

Calcule une métrique de régression pour évaluer la qualité des données prédites par rapport aux données réelles

```
double vector::RegressionMetric(
   const vector&           vector_true,   // vecteur de valeurs vraies
   ENUM_REGRESSION_METRIC  metric         // type de la métrique
   );

double matrix::RegressionMetric(
   const matrix&           matrix_true,   // matrice de valeurs vraies
   ENUM_REGRESSION_METRIC  metric         // type de la métrique
);

vector matrix::RegressionMetric(
   const matrix&           matrix_true,   // matrice de valeurs vraies
   ENUM_REGRESSION_METRIC  metric,        // type de la métrique
   int                     axis           // axe
   );
```

Paramètres

vector_true/matrix_true

[in] Vecteur ou matrice de valeurs vraies.

metric

[dans] Type de métrique de l'énumération ENUM_REGRESSION_METRIC.

axis

[in] Axe. 0 — axe horizontal, 1 — axe vertical.

Valeur de Retour

La métrique calculée qui évalue la qualité des données prédites par rapport aux données réelles.

Note

- REGRESSION_MAE — erreur absolue moyenne qui représente les différences absolues entre les valeurs prédites et les valeurs vraies correspondantes
- REGRESSION_MSE — erreur quadratique moyenne qui représente les différences au carré entre les valeurs prédites et les valeurs vraies correspondantes
- REGRESSION_RMSE — racine carrée de MSE
- REGRESSION_R2 - 1 — MSE(régression) / MSE(moyenne)
- REGRESSION_MAPE — MAE sous forme de pourcentage
- REGRESSION_MSPE — MSE sous forme de pourcentage
- REGRESSION_RMSLE — RMSE calculé sur une échelle logarithmique

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
   vector y_true = {3, -0.5, 2, 7};
   vector y_pred = {2.5, 0.0, 2, 8};
//---
   double mse=y_pred.RegressionMetric(y_true,REGRESSION_MSE);
   Print("mse=",mse);
//---
   double mae=y_pred.RegressionMetric(y_true,REGRESSION_MAE);
   Print("mae=",mae);
//---
   double r2=y_pred.RegressionMetric(y_true,REGRESSION_R2);
   Print("r2=",r2);

  /* Résultat
   mae=0.375
   mse=0.5
   r2=0.9486081370449679
   */
```

© 2000-2024, MetaQuotes Ltd.

ConfusionMatrix

Calcule la matrice de confusion. La méthode est appliquée au vecteur de valeurs prédites.

```
matrix vector::ConfusionMatrix(
   const vector&        vect_true      // vecteur de valeurs vraies
   );


matrix vector::ConfusionMatrix(
   const vector&        vect_true,     // vecteur de valeurs vraies
   uint                 label          // valeur de l'étiquette
   );
```

Paramètres

vect_true

[in] Vecteur de valeurs vraies.

label

[in] Valeur d'étiquette pour calculer la matrice de confusion.

Valeur de Retour

Matrice de confusion. Si aucune valeur d'étiquette n'est spécifiée, une matrice de confusion multi-classes est renvoyée, dans laquelle chaque étiquette est mise en correspondance individuellement avec les autres étiquettes. Si une valeur d'étiquette est spécifiée, une matrice 2 x 2 est renvoyée, dans laquelle l'étiquette spécifiée est considérée comme positive, tandis que toutes les autres étiquettes sont négatives (ovr, 'one vs rest', soit 'un contre le reste').

Note

La matrice de confusion C est telle que Cij est égal au nombre d'observations connues pour appartenir au groupe i et prédites comme appartenant au groupe j. Ainsi, dans la classification binaire, le nombre de vrais négatifs (TN) est C00, les faux négatifs (FN) sont C10, les vrais positifs (TP) sont C11 et les faux positifs (FP) sont C01.

En d'autres termes, la matrice peut être représentée graphiquement comme suit :

TN	FP
FN	TP

Les tailles du vecteur des valeurs vraies et du vecteur des valeurs prédites doivent être les mêmes.

Exemple :

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
```

© 2000-2024, MetaQuotes Ltd.

```
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
   matrix confusion=y_pred.ConfusionMatrix(y_true);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,0);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,1);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,2);
   Print(confusion);


/*
  [[3,0,0,0,0,0,0,0,0,0]
   [0,3,0,0,0,0,0,0,0,0]
   [0,0,1,0,1,0,0,1,0,0]
   [0,0,0,1,0,0,0,1,0,0]
   [0,0,1,0,3,0,0,0,0,1]
   [0,0,0,0,0,2,0,0,0,0]
   [1,0,0,0,0,1,1,0,0,0]
   [0,0,0,0,0,0,0,2,0,1]
   [0,0,1,0,0,0,0,0,0,1]
   [0,0,0,0,0,0,0,0,0,4]]
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
   [2,1]]
*/
```

© 2000-2024, MetaQuotes Ltd.

ConfusionMatrixMultilabel

Calcule la matrice de confusion pour chaque étiquette. La méthode est appliquée au vecteur de valeurs prédites.

```
uint vector::ConfusionMatrixMultiLabel(
   const vector&       vect_true,     // vecteur de valeurs vraies
   matrix&             confusions[]   // tableau de matrices de confusion calculées
   );
```

Paramètres

vect_true

[in] Vecteur de valeurs vraies.

confusions

[out] Un tableau de matrices 2 x 2 avec des matrices de confusion calculées pour chaque étiquette.

Valeur de Retour

Taille du tableau de matrices de confusion calculées. En cas d'échec, renvoie 0

Note

Le tableau de résultats peut être dynamique ou statique. Si le tableau est statique, sa taille ne doit pas être inférieure au nombre de classes.

Les tailles du vecteur des valeurs vraies et du vecteur des valeurs prédites doivent être les mêmes.

Exemple :

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
   matrix label_confusions[12];

   uint   res=y_pred.ConfusionMatrixMultiLabel(y_true,label_confusions);
   Print("res=",res,"  size=",label_confusions.Size());
   for(uint i=0; i<res; i++)
      Print(label_confusions[i]);


/*
  res=10  size=12
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
```

© 2000-2024, MetaQuotes Ltd.

```
    [2,1]]
   [[28,0]
    [1,1]]
   [[24,1]
    [2,3]]
   [[27,1]
    [0,2]]
   [[27,0]
    [2,1]]
   [[25,2]
    [1,2]]
   [[28,0]
    [2,0]]
   [[23,3]
    [0,4]]
*/
```

© 2000-2024, MetaQuotes Ltd.

ClassificationMetric

Calcule la métrique de classification pour évaluer la qualité des données prédites par rapport aux données vraies. La méthode est appliquée au vecteur de valeurs prédites.

```
vector vector::ClassificationMetric(
   const vector&               vect_true,      // vecteur de valeurs vraies
   ENUM_CLASSIFICATION_METRIC metric          // type de métrique
   );


vector vector::ClassificationMetric(
   const vector&               vect_true,      // vecteur de valeurs vraies
   ENUM_CLASSIFICATION_METRIC metric          // type de métrique
   ENUM_AVERAGE_MODE           mode            // mode de moyenne
   );
```

Paramètres

vect_true

[in] Vecteur de valeurs vraies.

metric

[in] Type de métrique, correspond à une valeur de l'énumération ENUM_CLASSIFICATION_METRIC. Des valeurs autres que CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION et CLASSIFICATION_ROC_AUC (utilisées dans la méthode ClassificationScore) sont appliquées.

mode

[in] Mode de moyenne de l'énumération ENUM_AVERAGE_MODE. Utilisé pour les métriques CLASSIFICATION_F1, CLASSIFICATION_JACCARD, CLASSIFICATION_PRECISION et CLASSIFICATION_RECALL.

Valeur de Retour

Un vecteur contenant la métrique calculée. Dans le cas du mode de moyenne AVERAGE_NONE, le vecteur contient des valeurs métriques pour chaque classe sans moyenne. (Par exemple, dans le cas d'une classification binaire, il s'agirait respectivement de deux métriques pour 'false' et 'true').

Remarque sur les modes de moyenne

AVERAGE_BINARY n'a de sens que pour la classification binaire.

AVERAGE_MICRO — calcule les métriques globalement en comptant le total des vrais positifs, des faux négatifs et des faux positifs.

AVERAGE_MACRO — calcule les métriques pour chaque étiquette et trouve leur moyenne non pondérée. Cela ne prend pas en compte le déséquilibre des étiquettes.

AVERAGE_WEIGHTED — calcule les métriques pour chaque étiquette et trouve leur moyenne pondérée par le support (le nombre d'instances vraies pour chaque étiquette). Cela modifie la 'macro'

© 2000-2024, MetaQuotes Ltd.

pour tenir compte du déséquilibre des étiquettes ; cela peut aboutir à un score F qui n'est pas entre précision et rappel.

Exemple :

```
vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

vector accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
Print("accuracy=",accuracy);
vector balanced=y_pred.ClassificationMetric(y_true,CLASSIFICATION_BALANCED_ACCURACY
Print("balanced=",balanced);
Print("");

vector f1_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MICRO)
Print("f1_micro=",f1_micro);
vector f1_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MACRO)
Print("f1_macro=",f1_macro);
vector f1_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_WEI
Print("f1_weighted=",f1_weighted);
vector f1_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_NONE);
Print("f1_none=",f1_none);
Print("");

vector jaccard_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVER
Print("jaccard_micro=",jaccard_micro);
vector jaccard_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVER
Print("jaccard_macro=",jaccard_macro);
vector jaccard_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,A
Print("jaccard_weighted=",jaccard_weighted);
vector jaccard_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVERA
Print("jaccard_none=",jaccard_none);
Print("");

vector precision_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,
Print("precision_micro=",precision_micro);
vector precision_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,
Print("precision_macro=",precision_macro);
vector precision_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISI
Print("precision_weighted=",precision_weighted);
vector precision_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,A
Print("precision_none=",precision_none);
Print("");

vector recall_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAG
Print("recall_micro=",recall_micro);
vector recall_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAG
Print("recall_macro=",recall_macro);
```

© 2000-2024, MetaQuotes Ltd.

```
   vector recall_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVE
   Print("recall_weighted=",recall_weighted);
   vector recall_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAGE
   Print("recall_none=",recall_none);
   Print("");

//--- binary classification
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};

   vector f1_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_F1,AVERAGE_
   Print("f1_bin=",f1_bin);
   vector jaccard_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_JACCAR
   Print("jaccard_bin=",jaccard_bin);
   vector precision_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_PREC
   Print("precision_bin=",precision_bin);
   vector recall_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_RECALL,
   Print("recall_bin=",recall_bin);


/*
  accuracy=[0.6666666666666666]
  balanced=[0.6433333333333333]

  f1_micro=[0.6666666666666666]
  f1_macro=[0.6122510822510823]
  f1_weighted=[0.632049062049062]
  f1_none=[0.8571428571428571,1,0.3333333333333333,0.6666666666666666,0.66666666666666

  jaccard_micro=[0.5]
  jaccard_macro=[0.4921428571428572]
  jaccard_weighted=[0.5056349206349205]
  jaccard_none=[0.75,1,0.2,0.5,0.5,0.6666666666666666,0.3333333333333333,0.4,0,0.57142

  precision_micro=[0.6666666666666666]
  precision_macro=[0.6571428571428571]
  precision_weighted=[0.6706349206349207]
  precision_none=[0.75,1,0.3333333333333333,1,0.75,0.6666666666666666,1,0.5,0,0.571428

  recall_micro=[0.6666666666666666]
  recall_macro=[0.6433333333333333]
  recall_weighted=[0.6666666666666666]
  recall_none=[1,1,0.3333333333333333,0.5,0.6,1,0.3333333333333333,0.6666666666666666,

  f1_bin=[0.4444444444444445]
  jaccard_bin=[0.2857142857142857]
  precision_bin=[0.5]
  recall_bin=[0.4]
*/
```

© 2000-2024, MetaQuotes Ltd.

ClassificationScore

Calcule la métrique de classification pour évaluer la qualité des données prédites par rapport aux données réelles.

Contrairement aux autres méthodes de la section Machine Learning, celle-ci s'applique au vecteur de valeurs vraies plutôt qu'au vecteur de valeurs prédites.

```
vector vector::ClassificationScore(
   const matrix&              pred_scores,   // matrice contenant la distribution de p
   ENUM_CLASSIFICATION_METRIC metric         // type de métrique
   ENUM_AVERAGE_MODE          mode           // mode de moyenne
   );


vector vector::ClassificationScore(
   const matrix&              pred_scores,   // matrice contenant la distribution de p
   ENUM_CLASSIFICATION_METRIC metric         // type de métrique
   int                        param          // paramètre supplémentaire
   );
```

Paramètres

pred_scores

[in] Une matrice contenant un ensemble de vecteurs horizontaux avec des probabilités pour chaque classe. Le nombre de lignes de la matrice doit correspondre à la taille du vecteur de valeurs vraies.

metric

[in] Type de métrique, correspond à une valeur de l'énumération ENUM_CLASSIFICATION_METRIC. Les valeurs CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION et CLASSIFICATION_ROC_AUC sont utilisées.

mode

[in] Mode de moyenne de l'énumération ENUM_AVERAGE_MODE. Utilisé pour les métriques CLASSIFICATION_AVERAGE_PRECISION et CLASSIFICATION_ROC_AUC.

param

[in] Dans le cas de la métrique CLASSIFICATION_TOP_K_ACCURACY, la valeur entière K doit être spécifiée au lieu du mode de moyenne.

Valeur de Retour

Un vecteur contenant la métrique calculée. Dans le cas du mode de moyenne AVERAGE_NONE, le vecteur contient des valeurs métriques pour chaque classe sans moyenne. (Par exemple, dans le cas d'une classification binaire, il s'agirait respectivement de deux métriques pour 'false' et 'true').

Remarque sur les modes de moyenne

AVERAGE_BINARY n'a de sens que pour la classification binaire.

© 2000-2024, MetaQuotes Ltd.

AVERAGE_MICRO — calcule les métriques globalement en comptant le total des vrais positifs, des faux négatifs et des faux positifs.

AVERAGE_MACRO — calcule les métriques pour chaque étiquette et trouve leur moyenne non pondérée. Cela ne prend pas en compte le déséquilibre des étiquettes.

AVERAGE_WEIGHTED — calcule les métriques pour chaque étiquette et trouve leur moyenne pondérée par le support (le nombre d'instances vraies pour chaque étiquette).

Note

En cas de classification binaire, nous pouvons saisir non seulement une matrice n x 2, où la première colonne contient des probabilités pour une étiquette négative et la deuxième colonne contient des probabilités pour une étiquette positive, mais également une matrice composée d'une colonne avec des probabilités positives. Cela est dû au fait que les modèles de classification binaire peuvent renvoyer soit 2 probabilités, soit 1 probabilité pour une étiquette positive.

Exemple :

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   //vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

//--- label scores          0         1         2         3         4         5
   matrix y_scores={{0.000109, 0.000186, 0.000449, 0.000052, 0.000002, 0.000022, 0.000
                    {0.000091, 0.081956, 0.916816, 0.001106, 0.000006, 0.000002, 0.000
                    {0.000108, 0.972863, 0.003600, 0.000021, 0.010479, 0.000015, 0.000
                    {0.925425, 0.000080, 0.002913, 0.000057, 0.000274, 0.000638, 0.063
                    {0.000060, 0.000126, 0.000006, 0.000000, 0.993513, 0.000000, 0.000
                    {0.000016, 0.982124, 0.000045, 0.000002, 0.008445, 0.000001, 0.000
                    {0.000000, 0.000040, 0.000001, 0.000000, 0.989395, 0.000167, 0.000
                    {0.000795, 0.002938, 0.023447, 0.007418, 0.021838, 0.002476, 0.000
                    {0.000091, 0.000226, 0.000038, 0.000007, 0.000048, 0.854910, 0.068
                    {0.000000, 0.000000, 0.000000, 0.000000, 0.003004, 0.000000, 0.000
                    {0.998856, 0.000009, 0.000976, 0.000002, 0.000000, 0.000013, 0.000
                    {0.000178, 0.000446, 0.000326, 0.000033, 0.000193, 0.000071, 0.998
                    {0.000005, 0.000016, 0.000153, 0.000045, 0.004110, 0.000012, 0.000
                    {0.994188, 0.000003, 0.002584, 0.000005, 0.000005, 0.000100, 0.000
                    {0.000173, 0.990569, 0.000792, 0.000040, 0.001798, 0.000035, 0.000
                    {0.000000, 0.000537, 0.000008, 0.005080, 0.000046, 0.992910, 0.000
                    {0.000127, 0.000003, 0.000003, 0.000000, 0.001583, 0.000000, 0.000
                    {0.000001, 0.000012, 0.000072, 0.000020, 0.000000, 0.000000, 0.000
                    {0.000020, 0.000105, 0.001139, 0.901343, 0.002132, 0.083873, 0.000
                    {0.000002, 0.000048, 0.000019, 0.000000, 0.999347, 0.000002, 0.000
                    {0.000059, 0.001344, 0.612502, 0.002749, 0.000229, 0.000678, 0.000
                    {0.000586, 0.000740, 0.001625, 0.000007, 0.269341, 0.000076, 0.016
                    {0.009547, 0.018055, 0.283795, 0.071079, 0.426074, 0.082335, 0.036
                    {0.002506, 0.002545, 0.001148, 0.005659, 0.020416, 0.000112, 0.006
                    {0.001263, 0.001769, 0.000293, 0.000011, 0.000302, 0.881768, 0.112
                    {0.002904, 0.002909, 0.013421, 0.001461, 0.007519, 0.001251, 0.000
                    {0.000055, 0.001080, 0.893158, 0.000000, 0.104492, 0.000159, 0.001
```

© 2000-2024, MetaQuotes Ltd.

```
                    {0.000344, 0.002693, 0.071184, 0.000262, 0.000001, 0.000003, 0.000
                    {0.001404, 0.009375, 0.002638, 0.229189, 0.000064, 0.000896, 0.007
                    {0.491140, 0.000125, 0.000024, 0.000302, 0.000038, 0.034947, 0.473

   vector top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,1);
   Print("top 1 accuracy score = ",top_k);
   top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top 2 accuracy score = ",top_k);
   vector y_true2={0, 1, 2, 2};
   matrix y_score2={{0.5, 0.2, 0.2},  // 0 est dans le top 2
                    {0.3, 0.4, 0.2},  // 1 est dans le top 2
                    {0.2, 0.4, 0.3},  // 2 est dans le top 2
                    {0.7, 0.2, 0.1}}; // 2 n'est pas dans le top 2
   top_k=y_true2.ClassificationScore(y_score2,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top k = ",top_k);
   Print("");

   vector ap_micro=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISIO
   Print("average precision score micro = ",ap_micro);
   vector ap_macro=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISIO
   Print("average precision score macro = ",ap_macro);
   vector ap_weighted=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECI
   Print("average precision score weighted = ",ap_weighted);
   vector ap_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISION
   Print("average precision score none = ",ap_none);
   Print("");

   vector area_micro=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAG
   Print("roc auc score micro = ",area_micro);
   vector area_macro=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAG
   Print("roc auc score macro = ",area_macro);
   vector area_weighted=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score weighted = ",area_weighted);
   vector area_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAGE
   Print("roc auc score none = ",area_none);
   Print("");

//--- binary classification
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};
   vector y_score_true={0.3,0.7,0.1,0.6,0.9,0.0,0.4,0.2,0.8};
   matrix y_score1_bin(y_score_true.Size(),1);
   y_score1_bin.Col(y_score_true,0);
   matrix y_scores_bin={{0.7, 0.3},
                        {0.3, 0.7},
                        {0.9, 0.1},
                        {0.4, 0.6},
                        {0.1, 0.9},
                        {1.0, 0.0},
```

```
                        {0.6, 0.4},
                        {0.8, 0.2},
                        {0.2, 0.8}};

   vector ap=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECIS
   Print("average precision score binary = ",ap);
   vector ap2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_AVERAGE_PRECI
   Print("average precision score binary = ",ap2);
   vector ap3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECI
   Print("average precision score none = ",ap3);
   Print("");

   vector area=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVER
   Print("roc auc score binary = ",area);
   vector area2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score binary = ",area2);
   vector area3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score none = ",area3);


/*
  top 1 accuracy score = [0.6666666666666666]
  top 2 accuracy score = [1]
  top k = [0.75]

  average precision score micro = [0.8513333333333333]
  average precision score macro = [0.9326666666666666]
  average precision score weighted = [0.9333333333333333]
  average precision score none = [1,1,0.7,1,0.9266666666666666,0.8333333333333333,1,0.

  roc auc score micro = [0.9839506172839506]
  roc auc score macro = [0.9892068783068803]
  roc auc score weighted = [0.9887354497354497]
  roc auc score none = [1,1,0.9506172839506173,1,0.984,0.9821428571428571,1,0.9753086

  average precision score binary = [0.7961904761904761]
  average precision score binary = [0.7961904761904761]
  average precision score none = [0.7678571428571428,0.7961904761904761]

  roc auc score binary = [0.7]
  roc auc score binary = [0.7]
  roc auc score none = [0.7,0.7]
*/
```

© 2000-2024, MetaQuotes Ltd.

PrecisionRecall

Compute values to construct a precision-recall curve. Similarly to ClassificationScore, this method is applied to the vector of true values.

```
bool vector::PrecisionRecall(
   const matrix&                 pred_scores,    // matrix containing the probability c
   const ENUM_ENUM_AVERAGE_MODE  mode            // averaging mode
   matrix&                       precision,      // calculated precision values for eac
   matrix&                       recall,         // calculated recall values for each t
   matrix&                       thresholds,     // threshold values sorted in descendi
   );
```

Parameters

pred_scores

[in] A matrix containing a set of horizontal vectors with probabilities for each class. The number of matrix rows must correspond to the size of the vector of true values.

mode

[in] Averaging mode from the ENUM_AVERAGE_MODE enumeration. Only AVERAGE_NONE, AVERAGE_BINARY and AVERAGE_MICRO are used.

precision

[out] A matrix with calculated precision curve values. If no averaging is applied (AVERAGE_NONE), the number of rows in the matrix corresponds to the number of model classes. The number of columns corresponds to the size of the vector of true values (or the number of rows in the probability distribution matrix pred_score). In the case of microaveraging, the number of rows in the matrix corresponds to the total number of threshold values, excluding duplicates.

recall

[out] A matrix with calculated recall curve values.

threshold

[out] Threshold matrix obtained by sorting the probability matrix

Note

See notes for the ClassificationScore method.

Example

An example of collecting statistics from the mnist.onnx model (99% accuracy).

```
//--- data for classification metrics
   vectorf y_true(images);
   vectorf y_pred(images);
   matrixf y_scores(images,10);
//--- input-output
   matrixf image(28,28);
```

© 2000-2024, MetaQuotes Ltd.

```
   vectorf result(10);

//--- testing
   for(int test=0; test<images; test++)
     {
      image=test_data[test].image;
      if(!OnnxRun(model,ONNX_DEFAULT,image,result))
        {
         Print("OnnxRun error ",GetLastError());
         break;
        }
      result.Activation(result,AF_SOFTMAX);
      //--- collect data
      y_true[test]=(float)test_data[test].label;
      y_pred[test]=(float)result.ArgMax();
      y_scores.Row(result,test);
     }    }
```

Accuracy calculation

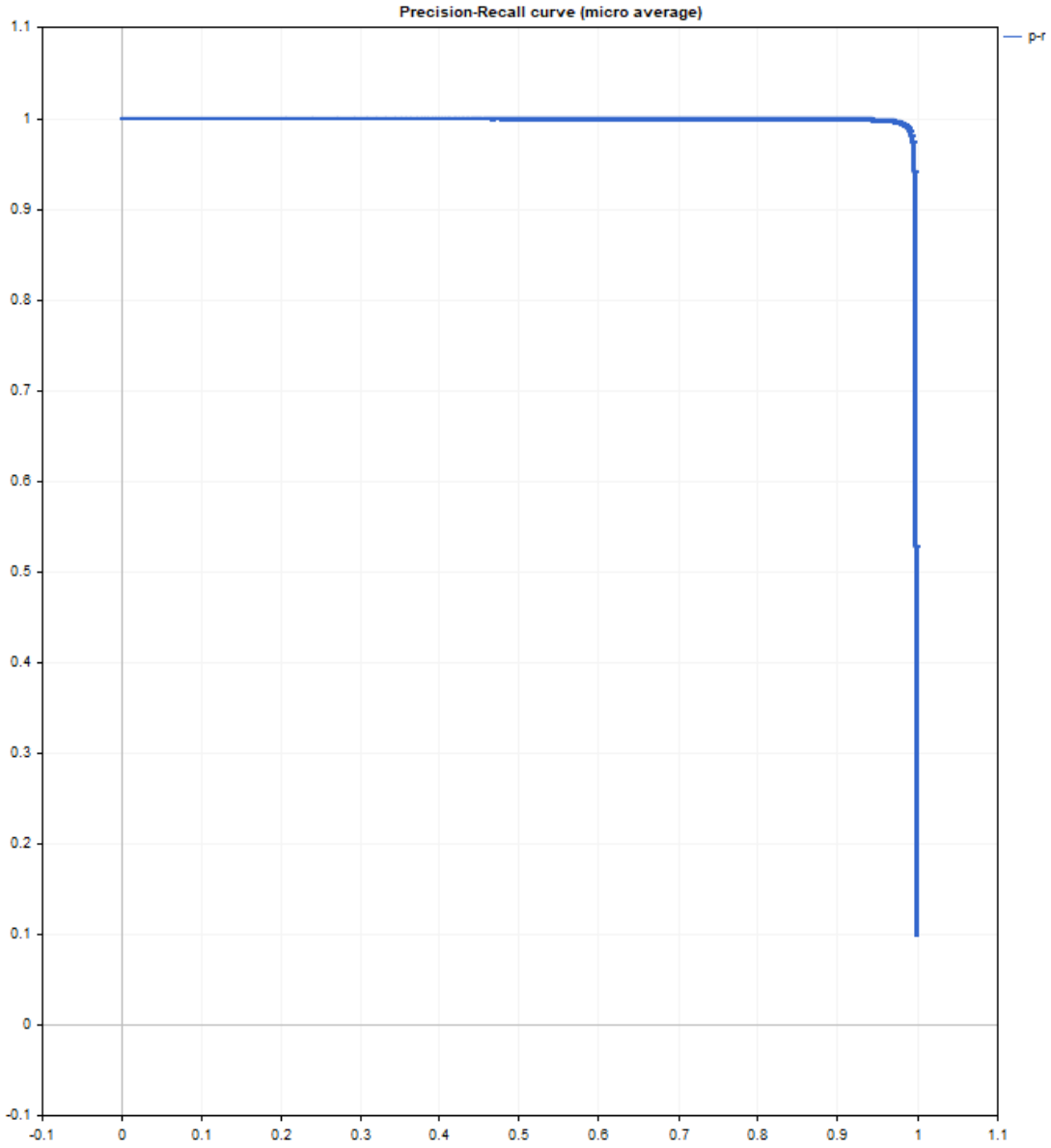
```
   vectorf accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
   PrintFormat("accuracy=%f",accuracy[0]);

accuracy=0.989000
```

An example of plotting precision-recall graphs, where precision values are plotted on the y-axis and recall values are plotted on the x-axis. Also precision and recall graphs are plotted separately, with threshold values plotted on the x-axis

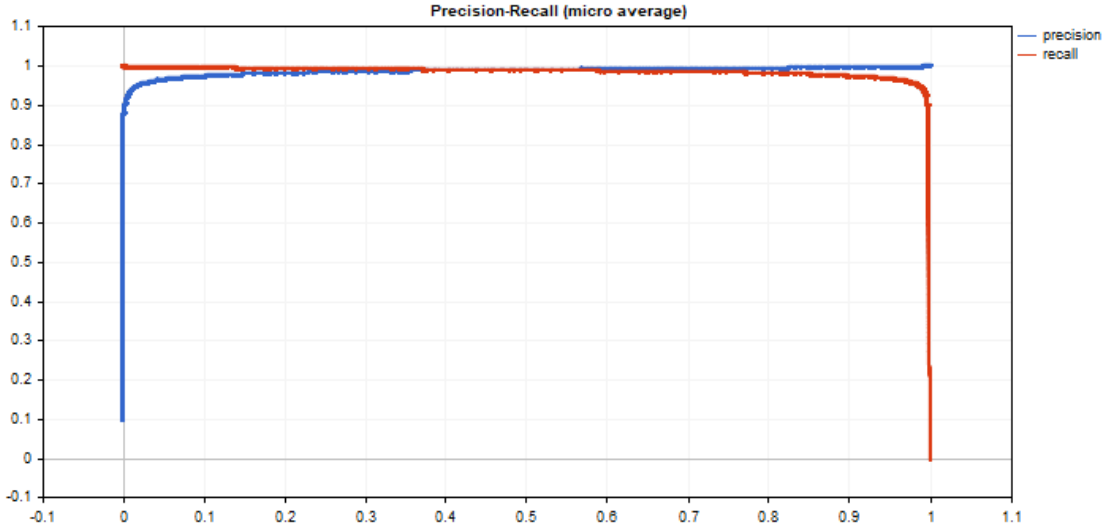
```
   if(y_true.PrecisionRecall(y_scores,AVERAGE_MICRO,mat_precision,mat_recall,mat_thres
     {
      double precision[],recall[],thres[];
      ArrayResize(precision,mat_thres.Cols());
      ArrayResize(recall,mat_thres.Cols());
      ArrayResize(thres,mat_thres.Cols());

      for(uint i=0; i<thres.Size(); i++)
        {
         precision[i]=mat_precision[0][i];
         recall[i]=mat_recall[0][i];
         thres[i]=mat_thres[0][i];
        }
      thres[0]=thres[1]+0.001;

      PlotCurve("Precision-Recall curve (micro average)","p-r","",recall,precision);
      Plot2Curves("Precision-Recall (micro average)","precision","recall",thres,precis
     }
```

© 2000-2024, MetaQuotes Ltd.

Resulting curves:



© 2000-2024, MetaQuotes Ltd.



© 2000-2024, MetaQuotes Ltd.

ReceiverOperatingCharacteristic

Compute values to construct the Receiver Operating Characteristic (ROC) curve. Similarly to ClassificationScore, this method is applied to the vector of true values.

```
bool vector::ReceiverOperatingCharacteristic(
   const matrix&                 pred_scores,    // matrix containing the probability c
   const ENUM_ENUM_AVERAGE_MODE  mode            // averaging mode
   matrix&                       fpr,            // calculated false positive rate valu
   matrix&                       tpr,            // calculated true positive rate value
   matrix&                       thresholds,     // threshold values sorted in descendi
   );
```

Parameters

pred_scores

[in] A matrix containing a set of horizontal vectors with probabilities for each class. The number of matrix rows must correspond to the size of the vector of true values.

mode

[in] Averaging mode from the ENUM_AVERAGE_MODE enumeration. Only AVERAGE_NONE, AVERAGE_BINARY and AVERAGE_MICRO are used.

fpr

[out] A matrix with calculated values of the false positive rate curve. If no averaging is applied (AVERAGE_NONE), the number of rows in the matrix corresponds to the number of model classes. The number of columns corresponds to the size of the vector of true values (or the number of rows in the probability distribution matrix pred_score). In the case of microaveraging, the number of rows in the matrix corresponds to the total number of threshold values, excluding duplicates.

tpr

[out] A matrix with calculated values of the true positive rate curve.

threshold

[out] Threshold matrix obtained by sorting the probability matrix

Note

See notes for the ClassificationScore method.

Example

An example of plotting ROC graphs, where tpr values are plotted on the y-axis and fpr values are plotted on the x-axis. Also fpr and tpr graphs are plotted separately, with threshold values plotted on the x-axis

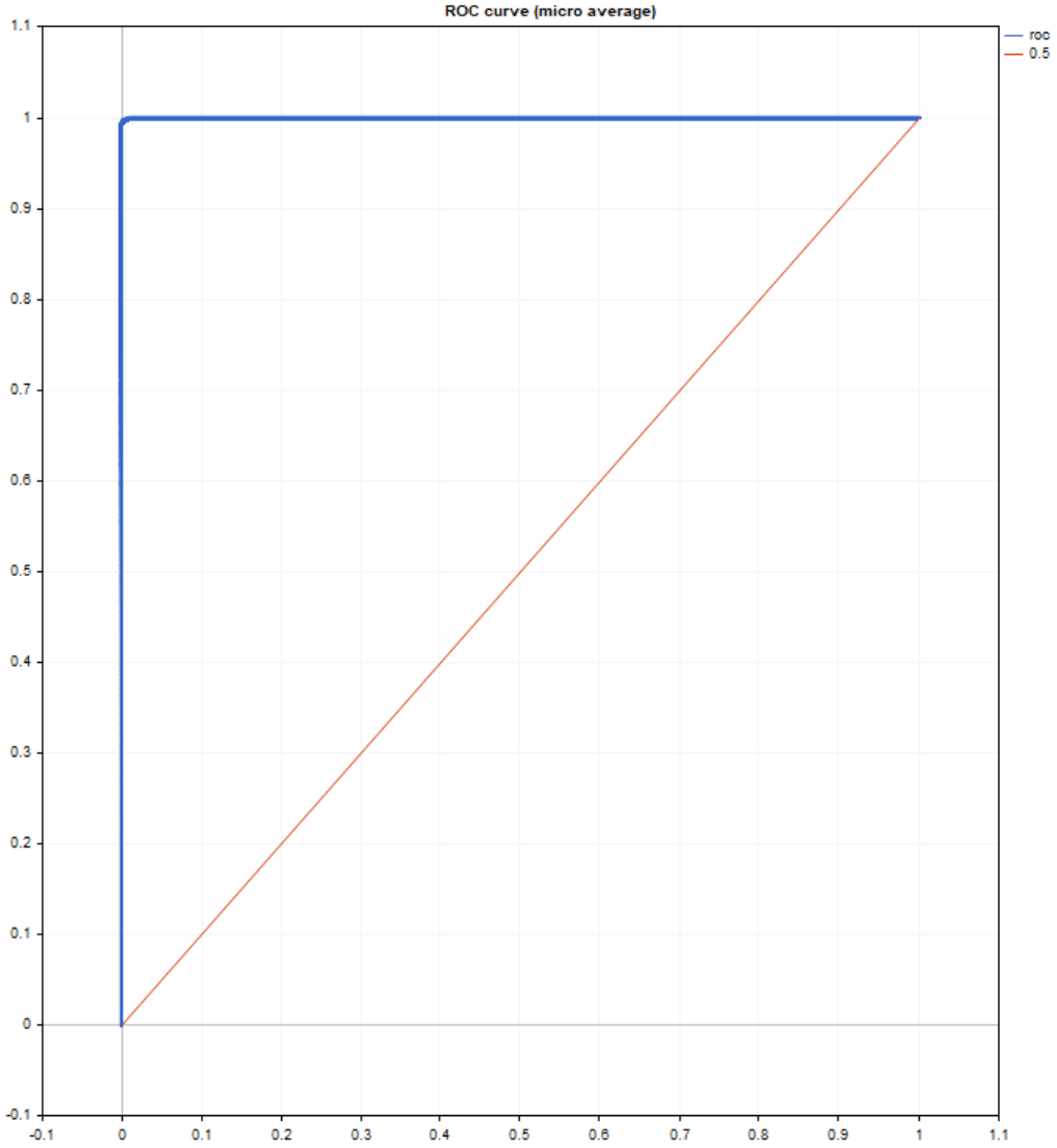
```
   matrixf mat_thres;
   matrixf mat_fpr;
   matrixf mat_tpr;
```

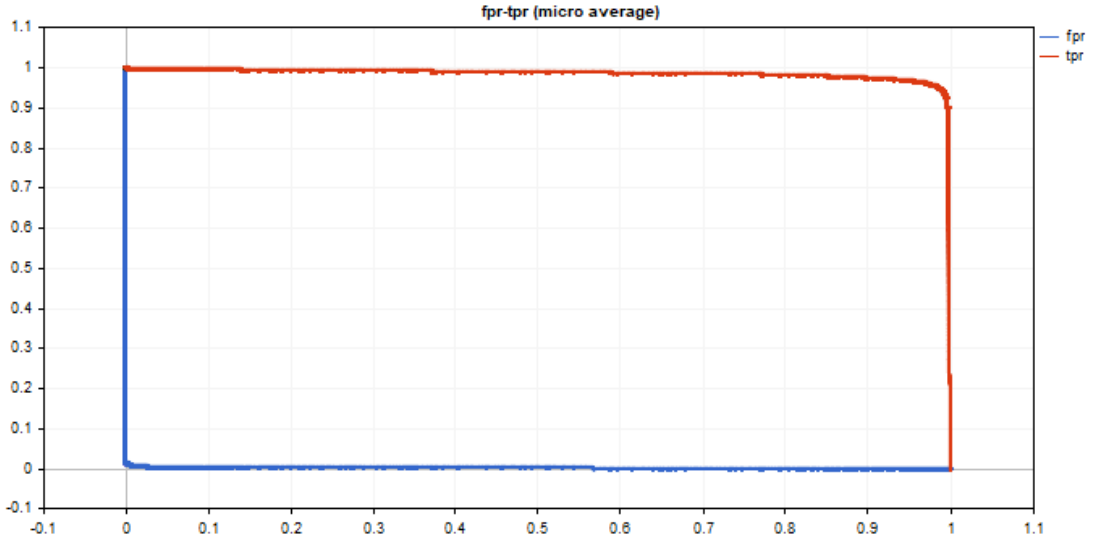
© 2000-2024, MetaQuotes Ltd.

```
    if(y_true.ReceiverOperatingCharacteristic(y_scores,AVERAGE_MICRO,mat_fpr,mat_tpr,ma
      {
       double fpr[],tpr[],thres[];
       ArrayResize(fpr,mat_thres.Cols());
       ArrayResize(tpr,mat_thres.Cols());
       ArrayResize(thres,mat_thres.Cols());

       for(uint i=0; i<fpr.Size(); i++)
         {
          fpr[i]=mat_fpr[0][i];
          tpr[i]=mat_tpr[0][i];
          thres[i]=mat_thres[0][i];
         }
       thres[0]=thres[1]+0.001;

       PlotCurve("ROC curve (micro average)","roc","0.5",fpr,tpr);
       Plot2Curves("fpr-tpr (micro average)","fpr","tpr",thres,fpr,tpr);
      }
```

Resulting curves:

© 2000-2024, MetaQuotes Ltd.





© 2000-2024, MetaQuotes Ltd.

The graph output code is simple and based on the <Graphics/Graphic.mqh> standard library.

The examples use the data of the mnist.onnx model. The code is presented in the PrecisionRecall method description.

ROC AUC is close to ideal.

```
roc auc score micro = [0.99991]
```

© 2000-2024, MetaQuotes Ltd.

La conversion des données

Le groupe des fonctions assurant la conversion des données d'un format aux données d'un autre format.

Il faut marquer spécialement la fonction NormalizeDouble(), qui assure l'exactitude demandée de la représentation du prix. Dans les opérations commerciales on ne peut pas utiliser les prix non normalisés, si l'exactitude dépasse les prix demandés par le serveur commercial même sur un signe.

Fonction	Action
CharToString	La conversion du code de symbole dans une chaîne d'un symbole
DoubleToString	La conversion de la valeur numérique à la chaîne de texte avec l'exactitude indiquée
EnumToString	La conversion de la valeur d'un transfert de n'importe quel type dans une chaîne
NormalizeDouble	L'arrondissement du nombre de la virgule flottante à l'exactitude indiquée
StringToDouble	La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type double
StringToInteger	La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type long
StringToTime	La conversion de la chaîne, contenant le temps et/ou la date dans le format "yyyy.mm.dd [hh:mi]", dans le nombre du type datetime
TimeToString	La conversion de la valeur, contenant le temps en secondes, passé après le 01.01.1970, à la chaîne du format "yyyy.mm.dd hh:mi"
IntegerToString	La conversion de la valeur du type entier à la chaîne de la longueur indiquée
ShortToString	La conversion du code de symbole (unicode)à la chaîne d'un symbole
ShortArrayToString	Copie la partie du tableau dans la chaîne
StringToShortArray	Copie la chaîne par un symbole à l'endroit spécifié du tableau du type ushort
CharArrayToString	Convertit le code de symbole (ansi) à la chaîne d'un symbole
StringToCharArray	Copie par un symbole la chaîne convertie de Unicode à ANSI à l'endroit spécifié du tableau du type uchar
CharArrayToStruct	Copie un tableau d'éléments de type uchar dans une structure POD
StructToCharArray	Copie la structure POD dans un tableau d'éléments de type uchar
ColorToARGB	Convertit le type color en type uint pour recevoir la représentation ARGB de la couleur.
ColorToString	Convertit la valeur de la couleur à la chaîne du type "R,G,B"

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
StringToColor	Convertit la chaîne du type "R,G,B" ou la chaîne, contenant le nom de la couleur, à la valeur du type color
StringFormat	Convertit le nombre à la chaîne conformément au format spécifié

Voir aussi

Utilisation de la page de code

© 2000-2024, MetaQuotes Ltd.

CharToString

La conversion du code de symbole dans une chaîne d'un symbole

```
string  CharToString(
   uchar  char_code      // code numérique du symbole
   );
```

Paramètres

char_code

[in] Le code du symbole ANSI.

La valeur rendue

La chaîne, contenant le symbole ANSI.

Voir aussi

StringToCharArray, ShortToString, StringGetCharacter

© 2000-2024, MetaQuotes Ltd.

CharArrayToString

Copie et convertit la partie du tableau du type uchar à la chaîne rendue.

```
string  CharArrayToString(
   uchar   array[],            // tableau
   int     start=0,            // position initiale dans le tableau
   int     count=-1            // nombre de symboles
   uint    codepage=CP_ACP     // page de code
   );
```

Paramètres

array[]

[in] Le tableau du type uchar.

start=0

[in] La position, par laquelle commence le copiage. Par défaut c'est 0.

count=-1

[in] Le nombre d'éléments de tableau pour le copiage. Définit la longueur de la chaîne de résultat. La valeur implicite est-1, qui signifie le copiage jusqu'à la fin du tableau ou jusqu'au 0 de terminal.

codepage=CP_ACP

[in] La valeur de la page de code. Pour les pages de code les plus utilisés on prévoit les constantes correspondantes.

La valeur rendue

La chaîne.

Voir aussi

StringToCharArray, ShortArrayToString, Utilisation de la page de code

© 2000-2024, MetaQuotes Ltd.

CharArrayToStruct

Copie un tableau d'éléments de type uchar dans une structure POD.

```
bool  CharArrayToStruct(
   void&          struct_object,     // structure
   const uchar&   char_array[],      // tableau
   uint           start_pos=0        // position de départ dans le tableau
   );
```

Paramètres

struct_object

[in] Référence sur n'importe quel type de structure POD (ne contenant que des types de données simples).

char_array[]

[in] tableau d'éléments de type uchar.

start_pos=0

[in] Position dans le tableau à partir de laquelle la copie des données commence.

Valeur de Retour

Retourne true en cas de succès, sinon false.

Voir également

StringToCharArray, ShortArrayToString, StructToCharArray, Utilisation d'une Page de Code (codepage), FileReadStruct, Unions (union), MathSwap

© 2000-2024, MetaQuotes Ltd.

StructToCharArray

Copie la structure POD dans un tableau d'éléments de type uchar.

```
bool  StructToCharArray(
   const void&  struct_object,      // structure
   uchar&       char_array[],       // tableau
   uint         start_pos=0         // position de départ dans le tableau
   );
```

Paramètres

struct_object

[in] Référence sur n'importe quel type de structure POD (ne contenant que des types de données simples).

char_array[]

[in] tableau d'éléments de type uchar.

start_pos=0

[in] Position dans le tableau à partir de laquelle les données copiées seront ajoutées.

Valeur de Retour

Retourne true en cas de succès, sinon false.

Note

Lors de la copie, le tableau dynamique est automatiquement étendu (ArrayResize) s'il n'y a pas assez d'espace. Si le tableau ne peut pas être étendu jusqu'à la valeur désirée la fonction retourne une erreur.

Voir également

StringToCharArray, ShortArrayToString,CharArrayToStruct, Utilisation d'une Page de Code (codepage), FileWriteStruct, Unions (union), MathSwap

© 2000-2024, MetaQuotes Ltd.

ColorToARGB

Convertit le type color en type uint pour recevoir la représentation ARGB de la couleur. Le format de la couleur ARGB est utilisé à la créationde la ressource graphique , de la sortie du texte et dans la classe de la bibliothèque standard CCanvas.

```
uint  ColorToARGB(
   color  clr,          // La couleur convertible dans le format color
   uchar  alpha=255     // Le canal alpha dirigeant la transparence de la couleur
   );
```

Réglages

clr

[in] La valeur de la couleur dans la variable du type color.

alpha

[in] La valeur du canal alpha, pour la réception de la couleur dans le formatARGB. Est spécifié par la valeur de 0 (la couleur du pixel appliqué ne change pas du tout l'affichage du pixel inférieur) jusqu'à 255 (la couleur est appliquée complètement et couvre la couleur du pixel inférieur). La transparence de la couleur en pourcentage est calculée comme (1-alpha/255) *100%, c'est-à-dire que plus la valeur du canal est moins grand plus la couleur est transparente.

La valeur rendue

La représentation de la couleur dans le format ARGB, où dans quatre octets du type uint sont prescrit par ordre les valeurs Alfa, Red, Green, Blue (le canal alpha, rouge, vert, bleu clair).

Note

Le format principal utilisé partout pour la description de la couleur du pixel sur l'écran de l'écran en infographie est RGB, où selon les noms des couleurs de base sont spécifiés les composants rouge (Red), vert (Green) et bleu (Blue) de la couleur. Chaque composant est décrit par un octet qui spécifie la densité de la couleur donnée dans l'intervalle de 0 jusqu'à 255 (de 0x00 jusqu'à 0XFF dans le format hexadécimal). Puisque la couleur blanche contient toutes les couleurs, il est décrit comme comme 0xFFFFFF, c'est-à-dire chacun de trois composants est présenté par la valeur maximal 0xFF.

Mais dans un certain nombre des tâches on demande l'indication de la transparence de la couleur pour décrire, comment l'image va se présenter, si la couleur avec une certaine transparence est appliquée sur cette image. Pour tels cas on introduit la notion du canal alpha (Alpha), qui est introduit comme le composant supplémentaire au format RGB. Le schéma du format ARGB est montré sur le dessin.

8								8								8								8							
Alpha								Red								Green								Blue							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Les valeurs ARGB sont indiquées d'habitude dans le format hexadécimal, où chaque paire des chiffres présente par ordre les valeurs des canaux Alpha, Red, Green et Blue. Par exemple, la couleur ARGB 80FFFF00 signifie la couleur jaune avec la transparence 50.2 %. Au début est 0x80, qui spécifie la valeur 50.2% du canal alpha, puisque c'est 50.2% de la valeur 0xFF; ensuite la première paire FF présente la valeur maximale du composant rouge; la paire suivante FF spécifie la même

© 2000-2024, MetaQuotes Ltd.

force du composant vert; et la dernière paire 00 présente la valeur minimale du composant bleu (l'absence de la couleur bleue). L'addition de la couleur verte et rouge donne la couleur jaune. Si le canal alpha n'est pas utilisé, l'enregistrement peut être réduite jusqu'à 6 chiffres RRGGBB, c'est pourquoi les valeurs du canal alpha sont stockées dans les octets entiers supérieurs comme uint.

En fonction du contexte les chiffres qui contiennent 16 symboles peuvent être enregistrés avec le préfixe '0x' ou '#', par exemple, 80FFFF00, ou 0x80FFFF00, ou #80FFFF00.

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- spécifions la transparence
   uchar alpha=0x55;  // la valeur 0x55 signifit 55/255=21.6 % de la transparence
   //--- déduisons la conversion au ARGB pour la couleur  clrBlue
   PrintFormat("0x%.8X - clrBlue",clrBlue);
   PrintFormat("0x%.8X - clrBlue ARGB with alpha=0x55 (transparency 21.6%%)",ColorToAR
   //---  déduisons la conversion au ARGB pour la couleur clrGreen
   PrintFormat("0x%.8X - clrGreen",clrGreen);
   PrintFormat("0x%.8X - clrGreen ARGB with alpha=0x55 (transparency 21.6%%)",ColorToA
   //---  déduisons la conversion au ARGB pour la couleur  clrRed
   PrintFormat("0x%.8X - clrRed",clrRed);
   PrintFormat("0x%.8X - clrRed ARGB with alpha=0x55 (transparency 21.6%%)",ColorToARG
  }
```

Voir aussi

Les ressources, ResourceCreate(), TextOut(), Le type color, Les types char, short, int et long

© 2000-2024, MetaQuotes Ltd.

ColorToString

Convertit la valeur de la couleur à la chaîne du type "R,G,B".

```
string  ColorToString(
   color   color_value,      // valeur de la couleur
   bool    color_name        // montrez le nom de couleur ou non
   );
```

Paramètres

color_value

[in] La valeur de la couleur dans la variable du type color.

color_name

[in] Le signe de la nécessité de rendre le nom de la couleur, dans le cas où la valeur de la couleur coïncide avec une des constantes de couleur prédéterminées.

La valeur rendue

La représentation de chaîne de la couleur dans l'aspect "R,G,B", où R, G et B représentent les constantes décimales transcrites à la chaîne avec la valeur de 0 jusqu'à 255. Si on a spécifié le paramètre color_name=true, on fait la tentative de convertir la valeur de la couleur au nom de la couleur.

Exemple:

```
   string clr=ColorToString(C'0,255,0'); // couleur verte
   Print(clr);

   clr=ColorToString(C'0,255,0',true);   // recevoir la constante de couleur
   Print(clr);
```

Voir aussi

StringToColor, ColorToARGB

© 2000-2024, MetaQuotes Ltd.

DoubleToString

La conversion de la valeur numérique à la chaîne de texte.

```
string  DoubleToString(
   double  value,       // nombre
   int     digits=8     // le nombre de chiffres après le point
   );
```

Paramètres

value

[in] La valeur avec la virgule flottante.

digits

[in] Le format de l'exactitude. Si la valeur digits est dans le domaine de 0 jusqu'à 16, on recevra la représentation de chaîne du nombre avec le nombre indiqué des signes après la virgule. Si la valeur digits est dans le domaine de -1 jusqu'au -16, on recevra la représentation de chaîne du nombre dans le format scientifique avec le nombre indiqué des signes après la virgule. Dans tous les autres cas la valeur de chaîne contiendra les 8 chiffres après le point.

La valeur rendue

La chaîne, contenant la représentation symbolique du nombre dans le format indiqué de l'exactitude.

Exemple:

```
   Print("DoubleToString(120.0+M_PI) : ",DoubleToString(120.0+M_PI));
   Print("DoubleToString(120.0+M_PI,16) : ",DoubleToString(120.0+M_PI,16));
   Print("DoubleToString(120.0+M_PI,-16) : ",DoubleToString(120.0+M_PI,-16));
   Print("DoubleToString(120.0+M_PI,-1) : ",DoubleToString(120.0+M_PI,-1));
   Print("DoubleToString(120.0+M_PI,-20) : ",DoubleToString(120.0+M_PI,-20));
```

Voir aussi

NormalizeDouble, StringToDouble

© 2000-2024, MetaQuotes Ltd.

EnumToString

La transformation de la valeur de l'énumération de n'importe quel type à la représentation de texte.

```
string  EnumToString(
   any_enum  value      // La valeur de l'énumération de tout type
   );
```

Paramètres

value

[in] La valeur de l'énumération de tout type.

La valeur rendue

Une chaîne contenant la représentation de texte de la valeur. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

La fonction peut définir dans la variable _LastError les valeurs suivantes des erreurs:

- ERR_INTERNAL_ERROR – l'erreur de l'environnement de l'exécution
- ERR_NOT_ENOUGH_MEMORY - pas assez de mémoire pour terminer l'opération
- ERR_INVALID_PARAMETER - Il est impossible de permettre le nom de la valeur de l'énumération

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
enum interval  // l'énumération des constantes nommées
  {
   month=1,      // l'intervalle dans un mois
   two_months,   // deux mois
   quarter,      // trois mois - le quartier
   halfyear=6,   // le semestre
   year=12,      // un an - 12 mois
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- établissons  l'intervalle temporaire qui est égal au mois
   interval period=month;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- établissons  l'intervalle temporaire qui est égal au quartier (trois mois))
   period=quarter;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- établissons  l'intervalle temporaire qui est égal à un an (12 mois))
   period=year;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- vérifions comment le type de l'ordre est déduit
   ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
   Print(EnumToString(type)+"="+IntegerToString(type));

//--- vérifions comment une valeur incorrecte est déduite
   type=WRONG_VALUE;
   Print(EnumToString(type)+"="+IntegerToString(type));

// Le résultat de l'exécution:
// month=1
// quarter=3
// year=12
// ORDER_TYPE_BUY=0
// ENUM_ORDER_TYPE::-1=-1
  }
```

Voir aussi

Les énumérations, Input les variables

© 2000-2024, MetaQuotes Ltd.

IntegerToString

La conversion de la valeur du type entier à la chaîne de la longueur indiquée et rend la chaîne reçue.

```
string  IntegerToString(
   long    number,              // nombre
   int     str_len=0,           // longueur de la chaîne sur la sortie
   ushort  fill_symbol=' '      // remplisseur
   );
```

Paramètres

number

[in] Le nombre pour la conversion.

str_len=0

[in] La longueur de la chaîne. Si la longueur de la chaîne reçue sera plus grande que la longueur indiquée, la chaîne ne se coupe pas. Si la longueur de la chaîne reçue sera plus petite, la chaîne reçue sera complétée par le symbole-le remplisseur.

fill_symbol=' '

[in] Le symbole-le remplisseur. C'est un espace par défaut.

La valeur rendue

La chaîne.

Voir aussi

StringToInteger

© 2000-2024, MetaQuotes Ltd.

ShortToString

La conversion du code de symbole (unicode)à la chaîne d'un symbole et rend la chaîne reçue.

```
string  ShortToString(
   ushort  symbol_code      // symbole
   );
```

Paramètres

symbol_code

[in] Le code de symbole. Au lieu du code du symbole on peut utiliser la chaîne littéral contenant le symbole, ou la chaîne avec le code hexadécimal de 2 bytes, correspondant au symbole du tableau Unicode.

La valeur rendue

La chaîne.

Voir aussi

StringToCharArray, CharToString, StringGetCharacter

© 2000-2024, MetaQuotes Ltd.

ShortArrayToString

Copie la partie du tableau dans la chaîne.

```
string  ShortArrayToString(
   ushort  array[],      // tableau
   int     start=0,      // position de départ dans le tableau
   int     count=-1      // nombre de symboles
   );
```

Paramètres

array[]

[in] Le tableau du type ushort (l'analogue du type wchar_t).

start=0

[in] La position, par laquelle commence le copiage. Par défaut c'est 0.

count=-1

[in] Le nombre d'éléments de tableau pour le copiage. Définit la longueur de la chaîne de résultat. La valeur implicite est-1, qui signifie le copiage jusqu'à la fin du tableau ou jusqu'au 0 de terminal.

La valeur rendue

La chaîne.

Voir aussi

StringToShortArray, CharArrayToString, Utilisation de la page de code

© 2000-2024, MetaQuotes Ltd.

TimeToString

La conversion de la valeur, contenant le temps en secondes, passé après le 01.01.1970, à la chaîne du format "yyyy.mm.dd hh:mi"

```
string  TimeToString(
   datetime  value,                          // nombre
   int       mode=TIME_DATE|TIME_MINUTES     // format de la sortie
   );
```

Paramètres

value

[in] Le temps en secondes de 00:00 le 1 janvier 1970.

mode=TIME_DATE|TIME_MINUTES

[in] Le mode supplémentaire de la sortie de données. Peut être un drapeau ou le drapeau combiné:
TIME_DATE reçoit le résultat en forme de " yyyy.mm.dd " ,
TIME_MINUTES reçoit le résultat en forme de " hh:mi " ,
TIME_SECONDS reçoit le résultat en forme de " hh:mi:ss ".

La valeur rendue

La chaîne.

Voir aussi

StringToTime, TimeToStruct

© 2000-2024, MetaQuotes Ltd.

NormalizeDouble

L'arrondissement du nombre de la virgule flottante à l'exactitude indiquée.

```
double  NormalizeDouble(
   double  value,       // nombre normalisé
   int     digits       // nombre de chiffres après le point
   );
```

Paramètres

value

[in] La valeur avec la virgule flottante.

digits

[in] Le format de l'exactitude, le nombre de chiffres après le point décimal (0-8).

La valeur rendue

La valeur du type double avec l'exactitude donnée.

Note

Les valeurs calculées StopLoss, TakeProfit, et ainsi que les valeurs de prix de l'ouverture des ordres remis, doivent être normalisées avec l'exactitude, dont la valeur peut être reçue par la fonction Digits().

Il faut avoir en vue que le nombre normalisé quand l'on déduit au Journal à l'aide de Print () peut contenir plus de signes après la virgule, que vous attendez. Par exemple,

```
   double a=76.671;              // le nombre normalisé avec 3 signes après la virgule
   Print("Print(76.671)=",a);    // le déduisons comme il est
   Print("DoubleToString(a,8)=",DoubleToString(a,8)); // déduisons avec l'exactitude s
```

déduira dans le terminal:

```
   DoubleToString(a,8)=76.67100000

   Print(76.671)=76.67100000000001
```

Exemple:

```
   double pi=M_PI;
   Print("pi = ",DoubleToString(pi,16));

   double pi_3=NormalizeDouble(M_PI,3);
   Print("NormalizeDouble(pi,3) = ",DoubleToString(pi_3,16))
   ;
   double pi_8=NormalizeDouble(M_PI,8);
   Print("NormalizeDouble(pi,8) = ",DoubleToString(pi_8,16));

   double pi_0=NormalizeDouble(M_PI,0);
   Print("NormalizeDouble(pi,0) = ",DoubleToString(pi_0,16));
```

© 2000-2024, MetaQuotes Ltd.

```
//   Résultat:
//   pi= 3.1415926535897931
//   NormalizeDouble(pi,3) = 3.1419999999999999
//   NormalizeDouble(pi,8) = 3.1415926499999998
//   NormalizeDouble(pi,0) = 3.0000000000000000
```

Voir aussi

DoubleToString, Les types matériels (double, float), La conformation des types

© 2000-2024, MetaQuotes Ltd.

StringToCharArray

Copie par un symbole la chaîne convertie de unicode à ansi à l'endroit spécifié du tableau du type uchar. La fonction rend le nombre d'éléments copiés.

```
int  StringToCharArray(
   string  text_string,          // chaîne-source
   uchar&  array[],              // tableau
   int     start=0,              // position de départ dans le tableau
   int     count=-1              // nombre de symboles
   uint    codepage=CP_ACP       // page de code
   );
```

Paramètres

text_string

[in] La chaîne pour le copiage.

array[]

[out] Le tableau du type uchar.

start=0

[in] La position, par laquelle commence le copiage. Par défaut c'est 0.

count=-1

[in] Le nombre d'éléments de tableau pour le copiage. Définit la longueur de la chaîne de résultat. La valeur implicite est-1, qui signifie le copiage jusqu'à la fin du tableau ou jusqu'au 0 de terminal. 0 de terminal sera aussi copié au tableau-récepteur, dans ce cas-là la grandeur du tableau dynamique peut être augmentée en cas de nécessité à la grandeur de la chaîne. Si la grandeur du tableau dynamique est plus grand que la longueur de la chaîne, la grandeur du tableau ne sera pas diminué.

codepage=CP_ACP

[in] La valeur de la page de code. Pour les pages de code les plus utilisés on prévoit les constantes correspondantes.

La valeur rendue

Le nombre d'éléments copiés.

Voir aussi

CharArrayToString, StringToShortArray, Utilisation de la page de code

© 2000-2024, MetaQuotes Ltd.

StringToColor

Convertit la chaîne du type "R,G,B" ou la chaîne, contenant le nom de la couleur, à la valeur du type color.

```
color  StringToColor(
   string  color_string      // représentation de la chaîne de couleur
   );
```

Paramètres

color_string

[in] La représentation de chaîne de la couleur comme "R,G,B" ou le nom d'un des couleurs Web prédéterminés.

La valeur rendue

La valeur de la couleur.

Exemple:

```
   color str_color=StringToColor("0,127,0");
   Print(str_color);
   Print((string)str_color);
//--- on change la couleur un peu
   str_color=StringToColor("0,128,0");
   Print(str_color);
   Print((string)str_color);
```

Voir aussi

ColorToString, ColorToARGB

© 2000-2024, MetaQuotes Ltd.

StringToDouble

La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type double.

```
double  StringToDouble(
   string  value      // chaîne
   );
```

Paramètres

value

[in] La chaîne, contenant la représentation symbolique du nombre.

La valeur rendue

La valeur du type double.

Voir aussi

NormalizeDouble, Les types matériels (double, float), La conformation des types

© 2000-2024, MetaQuotes Ltd.

StringToInteger

La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type long (entier).

```
long  StringToInteger(
   string  value      // chaîne
   );
```

Paramètres

value

[in] La chaîne, contenant le nombre.

La valeur rendue

La valeur du type long.

Voir aussi

IntegerToString, Les types matériels (double, float), La conformation des types

© 2000-2024, MetaQuotes Ltd.

StringToShortArray

Copie la chaîne par un symbole à l'endroit spécifié du tableau du type ushort ushort. La fonction rend le nombre d'éléments copiés.

```
int  StringToShortArray(
   string  text_string,      // chaîne-source
   ushort& array[],          // tableau
   int     start=0,          // position initiale dans le tableau
   int     count=-1          // nombre de symboles
   );
```

Paramètres

text_string

[in] La chaîne pour le copiage.

array[]

[out] Le tableau du type ushort (analogue du type wchar_t).

start=0

[in] La position, par laquelle commence le copiage. Par défaut c'est 0.

count=-1

[in] Le nombre d'éléments de tableau pour le copiage. Définit la longueur de la chaîne de résultat. La valeur implicite est-1, qui signifie le copiage jusqu'à la fin du tableau ou jusqu'au 0 de terminal. 0 de terminal sera aussi copié au tableau-récepteur, dans ce cas-là la grandeur du tableau dynamique peut être augmentée en cas de nécessité à la grandeur de la chaîne. Si la grandeur du tableau dynamique est plus grand que la longueur de la chaîne, la grandeur du tableau ne sera pas diminué.

La valeur rendue

Le nombre d'éléments copiés.

Voir aussi

ShortArrayToString, StringToCharArray, Utilisation de la page de code

© 2000-2024, MetaQuotes Ltd.

StringToTime

Transforme la chaîne de caracères contenant une heure et/ou unde date au format "aaaa.mm.jj [hh:mi]" en un nombre de type datetime.

```
datetime  StringToTime(
   const string  time_string      // chaîne de caractères contenant une date
   );
```

Parameters

time_string

[in] Chaîne de caractères dans un des formats suivant :

- "aaaa.mm.jj [hh:mi]"
- "aaaa.mm.jj [hh:mi:ss]"
- "aaaammjj [hh:mi:ss]"
- "aaaammjj [hhmiss]"
- "aaaa/mm/jj [hh:mi:ss]"
- "aaaa-mm-jj [hh:mi:ss]"

Valeur de Retour

Valeur de type datetime contenant le nombre de secondes passées depuis le 01.01.1970.

Note

Les caractères espace et tabulation situés entre la date et l'heure sont considérés comme un seul espace pour éviter tout traitement supplémentaire du paramètre *time_string* avant d'appeler StringToTime().

Voir également

TimeToString, TimeToStruct

© 2000-2024, MetaQuotes Ltd.

StringFormat

Formate les paramètres reçus et rend la chaîne.

```
string  StringFormat(
   string  format,      // chaîne avec la description du format
   ...     ...          // paramètres
   );
```

Paramètres

format

[in] La chaîne, contenant le moyen du formatage. Les règles du formatage sont les mêmes, comme pour la fonction PrintFormat

...

[in] Les paramètres divisés par la virgule.

La valeur rendue

La chaîne.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- строковые переменные
   string output_string;
   string temp_string;
   string format_string;
//--- подготовим заголовок спецификации
   temp_string=StringFormat("Спецификация контракта для %s:\n",_Symbol);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int digits=(int)SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   temp_string=StringFormat("   SYMBOL_DIGITS = %d (количество знаков после запятой)\n
                            digits);
   StringAdd(output_string,temp_string);
//--- вывод значения double с переменным количеством цифр после десятичной точки
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   format_string=StringFormat("   SYMBOL_POINT = %%.%df (значение одного пункта)\n",
                              digits);
   temp_string=StringFormat(format_string,point_value);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   temp_string=StringFormat("   SYMBOL_SPREAD = %d (текущий спред в пунктах)\n",
                            spread);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int min_stop=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_STOPS_LEVEL = %d (минимальный отступ в пу
                            min_stop);
   StringAdd(output_string,temp_string);
//--- вывод значения double без дробной части
   double contract_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_CONTRACT_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_CONTRACT_SIZE = %.f (размер контракта)\n"
                            contract_size);
   StringAdd(output_string,temp_string);
//--- вывод значения double с точностью по умолчанию
   double tick_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_TICK_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_TICK_SIZE = %f (минимальное изменение цен
                            tick_size);
   StringAdd(output_string,temp_string);
//--- определение способа начисления свопов
   int swap_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_SWAP_MODE);
   string str_swap_mode;
   switch(swap_mode)
     {
      case SYMBOL_SWAP_MODE_DISABLED: str_swap_mode="SYMBOL_SWAP_MODE_DISABLED (нет сп
      case SYMBOL_SWAP_MODE_POINTS: str_swap_mode="SYMBOL_SWAP_MODE_POINTS (в пунктах)
      case SYMBOL_SWAP_MODE_CURRENCY_SYMBOL: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_
      case SYMBOL_SWAP_MODE_CURRENCY_MARGIN: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_
      case SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY
      case SYMBOL_SWAP_MODE_INTEREST_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST
      case SYMBOL_SWAP_MODE_INTEREST_OPEN: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST_OP
      case SYMBOL_SWAP_MODE_REOPEN_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_CUR
      case SYMBOL_SWAP_MODE_REOPEN_BID: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_BID (пе
     }
//--- вывод значения string
   temp_string=StringFormat("   SYMBOL_SWAP_MODE = %s\n",
                            str_swap_mode);
   StringAdd(output_string,temp_string);
//--- вывод значения double с точностью по умолчанию
   double swap_long=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_LONG);
```

© 2000-2024, MetaQuotes Ltd.

```
   temp_string=StringFormat("   SYMBOL_SWAP_LONG = %f (своп на покупку)\n",
                            swap_long);
   StringAdd(output_string,temp_string);
//--- вывод значения double с точностью по умолчанию
   double swap_short=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_SHORT);
   temp_string=StringFormat("   SYMBOL_SWAP_SHORT = %f (своп на продажу)\n",
                            swap_short);
   StringAdd(output_string,temp_string);
//--- определение режима торговли
   int trade_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_MODE);
   string str_trade_mode;
   switch(trade_mode)
     {
      case SYMBOL_TRADE_MODE_DISABLED: str_trade_mode="SYMBOL_TRADE_MODE_DISABLED (тор
      case SYMBOL_TRADE_MODE_LONGONLY: str_trade_mode="SYMBOL_TRADE_MODE_LONGONLY (раз
      case SYMBOL_TRADE_MODE_SHORTONLY: str_trade_mode="SYMBOL_TRADE_MODE_SHORTONLY (р
      case SYMBOL_TRADE_MODE_CLOSEONLY: str_trade_mode="SYMBOL_TRADE_MODE_CLOSEONLY (р
      case SYMBOL_TRADE_MODE_FULL: str_trade_mode="SYMBOL_TRADE_MODE_FULL (нет огранич
     }
//--- вывод значения string
   temp_string=StringFormat("   SYMBOL_TRADE_MODE = %s\n",
                            str_trade_mode);
   StringAdd(output_string,temp_string);
//--- вывод значения double в компактном виде
   double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   temp_string=StringFormat("   SYMBOL_VOLUME_MIN = %g (минимальный объем сделки)\n",v
   StringAdd(output_string,temp_string);
//--- вывод значения double в компактном виде
   double volume_step=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_STEP);
   temp_string=StringFormat("   SYMBOL_VOLUME_STEP = %g (минимальный шаг изменения об
   StringAdd(output_string,temp_string);
//--- вывод значения double в компактном виде
   double volume_max=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MAX);
   temp_string=StringFormat("   SYMBOL_VOLUME_MAX = %g (максимальный объем сделки)\n",
   StringAdd(output_string,temp_string);
//--- определение способа вычисления величины залоговых средств
   int calc_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_CALC_MODE);
   string str_calc_mode;
   switch(calc_mode)
     {
      case SYMBOL_CALC_MODE_FOREX:str_calc_mode="SYMBOL_CALC_MODE_FOREX (Forex)";break
      case SYMBOL_CALC_MODE_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_FUTURES (фьючерсы)
      case SYMBOL_CALC_MODE_CFD:str_calc_mode="SYMBOL_CALC_MODE_CFD (CFD)";break;
      case SYMBOL_CALC_MODE_CFDINDEX:str_calc_mode="SYMBOL_CALC_MODE_CFDINDEX (CFD на
      case SYMBOL_CALC_MODE_CFDLEVERAGE:str_calc_mode="SYMBOL_CALC_MODE_CFDLEVERAGE (C
      case SYMBOL_CALC_MODE_EXCH_STOCKS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_STOCKS (т
      case SYMBOL_CALC_MODE_EXCH_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FUTURES
      case SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FU
     }
//--- вывод значения string
   temp_string=StringFormat("   SYMBOL_TRADE_CALC_MODE = %s\n",
                            str_calc_mode);
   StringAdd(output_string,temp_string);
//--- вывод значения double с 2 цифрами после десятичной точки
   double margin_initial=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_INITIAL);
   temp_string=StringFormat("   SYMBOL_MARGIN_INITIAL = %.2f (начальная маржа)\n",
                            margin_initial);
   StringAdd(output_string,temp_string);
//--- вывод значения double с 2 цифрами после десятичной точки
   double margin_maintenance=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_MAINTENANCE);
   temp_string=StringFormat("   SYMBOL_MARGIN_MAINTENANCE = %.2f (поддерживающая марж
```

© 2000-2024, MetaQuotes Ltd.

```
                              margin_maintenance);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int freeze_level=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_FREEZE_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_FREEZE_LEVEL = %d (дистанция заморозки ор
                              freeze_level);
   StringAdd(output_string,temp_string);
   Print(output_string);
   Comment(output_string);
/* результат выполнения
   Спецификация контракта для EURUSD:
     SYMBOL_DIGITS = 5 (количество знаков после запятой)
     SYMBOL_POINT = 0.00001 (значение одного пункта)
     SYMBOL_SPREAD = 10 (текущий спред в пунктах)
     SYMBOL_TRADE_STOPS_LEVEL = 18 (минимальный отступ в пунктах для стоп-ордеров)
     SYMBOL_TRADE_CONTRACT_SIZE = 100000 (размер контракта)
     SYMBOL_TRADE_TICK_SIZE = 0.000010 (минимальное изменение цены)
     SYMBOL_SWAP_MODE = SYMBOL_SWAP_MODE_POINTS (в пунктах)
     SYMBOL_SWAP_LONG = -0.700000 (своп на покупку)
     SYMBOL_SWAP_SHORT = -1.000000 (своп на продажу)
     SYMBOL_TRADE_MODE = SYMBOL_TRADE_MODE_FULL (нет ограничений на торговые операции)
     SYMBOL_VOLUME_MIN = 0.01 (минимальный объем сделки)
     SYMBOL_VOLUME_STEP = 0.01 (минимальный шаг изменения объема сделки)
     SYMBOL_VOLUME_MAX = 500 (максимальный объем сделки)
     SYMBOL_TRADE_CALC_MODE = SYMBOL_CALC_MODE_FOREX (Forex)
     SYMBOL_MARGIN_INITIAL = 0.00 (начальная маржа)
     SYMBOL_MARGIN_MAINTENANCE = 0.00 (поддерживающая маржа)
     SYMBOL_TRADE_FREEZE_LEVEL = 0 (дистанция заморозки операций в пунктах)
*/
  }
```

Voir aussi

PrintFormat, DoubleToString,ColorToString, TimeToString

© 2000-2024, MetaQuotes Ltd.

Les fonctions mathématiques

Un ensemble des fonctions mathématiques et trigonométriques.

Les fonctions mathématiques ont été conçues à l'origine pour effectuer des opérations spécifiques sur des valeurs scalaires. À partir de cette construction, la plupart des fonctions peuvent être appliquées aux matrices et aux vecteurs. Parmi celles-ci, nous trouvons MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh et MathTanh . Ces opérations impliquent une manipulation élément par élément des matrices ou des vecteurs. Exemple :

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

Pour MathMod et MathPow, le deuxième élément peut être soit un scalaire, soit une matrice/vecteur de la bonne taille.

Fonction	Action
MathAbs	Retourne la valeur absolue (la valeur selon le module) du nombre transmis à elle
MathArccos	Retourne la valeur d'arc cosinus x aux radians
MathArcsin	Retourne la valeur d'arc sinus x aux radians
MathArctan	Retourne l'arc tangent x aux radians
MathArctan2	Retourne l'angle (en radians) dont la tangente est le rapport de 2 nombres spécifiés.
MathClassify	Renvoie le type d'un nombre réel
MathCeil	Retourne la valeur entière numérique la plus proche par dessus
MathCos	Retourne le cosinus du nombre
MathExp	Retourne l'exposant au nombre
MathFloor	Retourne la valeur entière numérique la plus proche par dessous
MathLog	Retourne le logarithme naturel

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
MathLog10	Retourne le logarithme du nombre selon le fondement 10
MathMax	Retourne la valeur maximal des deux valeurs numériques
MathMin	Retourne la valeur minimal des deux valeurs numériques
MathMod	Retourne le reste réel après la division de deux nombres
MathPow	Elève le fondement à la puissance indiquée
MathRand	Retourne le nombre entier pseudo-accidentelle dans le domaine de 0 jusqu'à 32767
MathRound	Arrondit le nombre jusqu'à l'entier plus proche
MathSin	Retourne le sinus du nombre
MathSqrt	Retourne la racine carrée
MathSrand	Établit l'état initial du générateur des nombres entiers quasi-aléatoires
MathTan	Retourne la tangent du nombre
MathIsValidNumber	Vérifie l'exactitude d'un nombre réel
MathExpm1	Retourne la valeur de l'expression MathExp(x)-1
MathLog1p	Retourne la valeur de l'expression MathLog(1+x)
MathArccosh	Retourne l'arc cosinus hyperbolique
MathArcsinh	Retourne l'arc sinus hyperbolique
MathArctanh	Retourne l'arc tangente hyperbolique
MathCosh	Retourne le cosinus hyperbolique
MathSinh	Retourne le sinus hyperbolique
MathTanh	Retourne la tangente hyperbolique
MathSwap	Change l'ordre des octets dans la valeur des types ushort/uint/ushort

© 2000-2024, MetaQuotes Ltd.

MathAbs

Rend la valeur absolue (la valeur selon le module) du nombre transmis à elle.

```
double  MathAbs(
   double  value      // nombre
   );
```

Paramètres

value

[in] La valeur numérique.

La valeur rendue

La valeur du type double, est plus que ou égal à zéro.

Note

On peut utiliser la fonction fabs() au lieu de la fonction MathAbs().

© 2000-2024, MetaQuotes Ltd.

MathArccos

Rend la valeur d'arc cosinus x dans le domaine 0 au π dans les radians.

```
double  MathArccos(
   double  val     // -1<val<1
   );
```

Paramètres

val

[in] La valeur val est entre -1 et 1, dont l'arc cosinus doit être calculé.

La valeur rendue

L'arc cosinus du nombre est dans les radians. Si val est moins que -1 ou plus que 1, la fonction rend NaN (une valeur indéterminée).

Note

On peut utiliser la fonction acos() au lieu de la fonction MathArccos().

Voir aussi

Les types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathArcsin

Rend l'arc sinus x dans le domaine de -π/2 jusqu'au π/2 des radians.

```
double  MathArcsin(
   double  val      // -1<value<1
   );
```

Paramètres

val

[in] La valeur val est entre -1 et 1, dont l'arc sinus doit être calculé.

La valeur rendue

L'arc sinus du nombre val dans les radians dans le domaine de -π/2 jusqu'au π/2 des radians. Si val est moins que -1 ou plus que 1, la fonction rend NaN (une valeur indéterminée).

Note

On peut utiliser la fonction asin() au lieu de la fonction MathArcsin().

Voir aussi

Les types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathArctan

Rend l'arc tangent x. Si x est égal au 0, la fonction rend 0.

```
double  MathArctan(
   double  value      // tangent
   );
```

Paramètres

value

[in] Le nombre qui représente la tangente.

La valeur rendue

MathArctan rend la valeur dans le domaine de -π/2 jusqu'au les radians π/2.

Note

On peut utiliser la fonction atan() au lieu de la fonction MathArctan().

© 2000-2024, MetaQuotes Ltd.

MathArctan2

Retourne l'angle (en radians) dont la tangente est le rapport de 2 nombres spécifiés.

```
double MathArctan2(
 double y     // La coordonnée y d'un point
 double x     // La coordonnée x d'un point
  );
```

Paramètres

y

[in] Valeur de la coordonnée Y.

x

[in] Valeur de la coordonnée X.

Valeur de Retour

MathArctan2 retourne un angle, θ, dans l'intervalle -π à π radians, afin que MathTan(θ) = y/x.

Veuillez noter ce qui suit :

- Pour (x, y) dans le quadrant 1, 0 < θ < π/2
- Pour (x, y) dans le quadrant 2, π/2 < θ ≤ π
- Pour (x, y) dans le quadrant 3, -π < θ < -π/2
- Pour (x, y) dans le quadrant 4, -π/2 < θ < 0

Pour les points situés sur les limites des quadrants, la valeur de retour est la suivante :

- Si y = 0 et x n'est pas négatif, θ = 0.
- Si y = 0 et x est négatif, θ = π.
- Si y est positif et x = 0, θ = π/2.
- Si y est négatif et x = 0, θ = -π/2.
- Si y = 0 et x = 0, θ = 0.

Note

Au lieu d'utiliser la fonction MathArctan2(), vous pouvez utiliser la fonction atan2().

© 2000-2024, MetaQuotes Ltd.

MathClassify

Détermine le type d'un nombre réel et renvoie un résultat sous forme de valeur de l'énumération ENUM_FP_CLASS

```
ENUM_FP_CLASS  MathClassify(
   double  value      // nombre réel
   );
```

Paramètres

value

[in] Le nombre réel à vérifier

Valeur de Retour

Une valeur de l'énumération ENUM_FP_CLASS

ENUM_FP_CLASS

ID	Description
FP_SUBNORMAL	Un nombre sous-normal plus proche de zéro que le plus petit nombre normal représentable DBL_MIN (2.2250738585072014e-308)
FP_NORMAL	Un nombre normal compris entre 2,2250738585072014e-308 et 1,7976931348623158e+308
FP_ZERO	Un zéro positif ou négatif
FP_INFINITE	Un nombre qui ne peut pas être représenté par le type approprié, l'infini positif ou négatif
FP_NAN	Pas un chiffre.

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- test NaN
  double nan=double("nan");
  PrintFormat("Test NaN: %G is %s, MathIsValidNumber(NaN)=%s",
              nan,
              EnumToString(MathClassify(nan)),
              (string)MathIsValidNumber(nan));
//--- test infinity
  double inf=double("inf");
  PrintFormat("Test Inf: %G is %s, MathIsValidNumber(inf)=%s",
```

© 2000-2024, MetaQuotes Ltd.

```
                inf,
                EnumToString(MathClassify(inf)),
                (string)MathIsValidNumber(inf));
//--- test normal value
  double normal=1.2345e6;
  PrintFormat("Test Normal: %G is %s, MathIsValidNumber(normal)=%s",
                normal,
                EnumToString(MathClassify(normal)),
                (string)MathIsValidNumber(normal));
//--- test subnormal value
  double sub_normal=DBL_MIN/2.0;
  PrintFormat("Test Subnormal: %G is %s, MathIsValidNumber(sub_normal)=%s",
                sub_normal,
                EnumToString(MathClassify(sub_normal)),
                (string)MathIsValidNumber(sub_normal));
//--- test zero value
  double zero=0.0/(-1);
  PrintFormat("Test Zero: %G is %s, MathIsValidNumber(zero)=%s",
                zero,
                EnumToString(MathClassify(zero)),
                (string)MathIsValidNumber(zero));
 }
 /*
 Result:
   Test NaN: NAN is FP_NAN, MathIsValidNumber(NaN)=false
   Test Inf: INF is FP_INFINITE, MathIsValidNumber(inf)=false
   Test Normal: 1.2345E+06 is FP_NORMAL, MathIsValidNumber(normal)=true
   Test Subnormal: 1.11254E-308 is FP_SUBNORMAL, MathIsValidNumber(sub_normal)=true
   Test Zero: -0 is FP_ZERO, MathIsValidNumber(zero)=true
*/
//+------------------------------------------------------------------+
```

Voir aussi

Types réels (double, flottant), MathIsValidNumber

© 2000-2024, MetaQuotes Ltd.

MathCeil

Rend la valeur entière numérique la plus proche par dessus.

```
double  MathCeil(
   double  val      // nombre
   );
```

Paramètres

val

[in] La valeur numérique.

La valeur rendue

La valeur numérique, représentant le plus petit nombre entier, qui est plus ou égal à val.

Note

On peut utiliser la fonction ceil() au lieu de la fonction MathCeil().

© 2000-2024, MetaQuotes Ltd.

MathCos

La fonction rend le cosinus du nombre.

```
double  MathCos(
   double  value      // nombre
   );
```

Paramètres

value

[in] L'angle est aux radians.

La valeur rendue

La valeur du type double dans le domine de -1 jusqu'au 1.

Note

On peut utiliser la fonction cos()au lieu de la fonction MathCos().

© 2000-2024, MetaQuotes Ltd.

MathExp

La fonction revient la valeur du nombre e à la puissance d.

```
double  MathExp(
   double  value      // puissance pour le nombre e
   );
```

Paramètres

value

[in] Le nombre définissant la puissance.

La valeur rendue

Le nombre du type double. Au débordement, la fonction rend INF (l'infinité), en cas de la perte de l'ordre de MathExp rend 0.

Note

On peut utiliser la fonction exp()au lieu de la fonction MathExp().

Voir aussi

Les types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathFloor

Rend la valeur entière numérique la plus proche par dessous.

```
double  MathFloor(
   double  val     // nombre
   );
```

Paramètres

val

[in] La valeur numérique.

La valeur rendue

La valeur numérique, représentant le plus grand nombre entier, qui est plus ou égal à val.

Note

On peut utiliser la fonction floor() au lieu de la fonction MathFloor().

© 2000-2024, MetaQuotes Ltd.

MathLog

Rend le logarithme naturel.

```
double  MathLog(
   double  val      // nombre pour prendre le logarithme
   );
```

Paramètres

val

[in] La valeur dont le logarithme doit être calculé.

La valeur rendue

Le logarithme naturel est val en cas du succès. Si la valeur val est negative, la fonction rend NaN (une valeur indéterminée). Si val est égal au 0, la fonction rend INF (l'infinité).

Note

On peut utiliser la fonction log() au lieu de la fonction MathLog().

Voir aussi

Les types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathLog

Rend le logarithme du nombre selon le fondement 10.

```
double  MathLog10(
   double  val      // nombre pour prendre le logarithme
   );
```

Paramètres

val

[in] La valeur dont le logarithme décimal doit être calculé.

La valeur rendue

le logarithme décimal est val en cas du succès. Si la valeur val est négative, la fonction rend NaN (une valeur indéterminée). Si val est égal au 0, la fonction rend INF (l'infinité).

Note

On peut utiliser la fonction log10() au lieu de la fonction MathLog10().

Voir aussi

Les types réels(double, float)

© 2000-2024, MetaQuotes Ltd.

MathMax

Rend la valeur maximal des deux valeurs numériques.

```
double  MathMax(
   double  value1,      // première valeur
   double  value2       // deuxième valeur
   );
```

Paramètres

value1

[in] La première valeur numérique.

value2

[in] La deuxième valeur numérique.

La valeur rendue

Le plus grand de deux nombres.

Note

On peut utiliser la fonction fmax()au lieu de la fonction MathMax(). Les fonctions fmax(), fmin(), MathMax(), MathMin() peuvent travailler avec les types entiers sans transformation au type double.

Si on transmet les paramètres des types différents dans la fonction, le paramètre du type bas est amené automatiquement au type principal. Le type de la valeur rendue correspond au type principal.

Si on transmet les données d'un type, aucune transformation n'est pas produite.

© 2000-2024, MetaQuotes Ltd.

MathMin

Rend la valeur minimal des deux valeurs numériques.

```
double  MathMin(
   double  value1,      // première valeur
   double  value2       // deuxième valeur
   );
```

Paramètres

value1

[in] La première valeur numérique.

value2

[in] La deuxième valeur numérique.

La valeur rendue

Le plus petit de deux nombres.

Note

On peut utiliser la fonction fmin()au lieu de la fonction MathMin(). Les fonctions fmax(), fmin(), MathMax(), MathMin() peuvent travailler avec les types entiers sans transformation au type double.

Si on transmet les paramètres des types différents dans la fonction, le paramètre du type bas est amené automatiquement au type principal. Le type de la valeur rendue correspond au type principal.

Si on transmet les données d'un type, aucune transformation n'est pas produite.

© 2000-2024, MetaQuotes Ltd.

MathMod

Rend le reste réel après la division de deux nombres.

```
double  MathMod(
   double  value,       // dividende
   double  value2       // diviseur
   );
```

Paramètres

value

[in] La valeur du dividence.

value2

[in] La valeur du diviseur.

La valeur rendue

La fonction MathMod calcule le reste réel f de val / y de manière que val = i * y + f , où i est le nombre entier, f a le même signe que val, et la valeur absolue f est moins que la valeur absolue y.

Note

On peut utiliser la fonction fmod()au lieu de la fonction MathMod().

© 2000-2024, MetaQuotes Ltd.

MathPow

Elève le fondement à la puissance indiquée.

```
double  MathPow(
   double  base,          // fondement
   double  exponent       // valeur d'exposant
   );
```

Paramètres

base

[in] Le fondement.

exponent

[in] La valeur de la puissance.

La valeur rendue

La valeur du fondement, élevé à la puissance indiquée.

Note

On peut utiliser la fonction pow()au lieu de la fonction MathPow().

© 2000-2024, MetaQuotes Ltd.

MathRand

Rend le nombre entier pseudo-accidentelle dans le domaine de 0 jusqu'à 32767.

```
int  MathRand();
```

La valeur rendue

Le nombre entier dans le domaine de 0 jusqu'au 32767.

Note

Avant le premier appel de la fonction il est nécessaire d'utiliser la fonction MathSrand, pour transmettre le générateur des nombres pseudo-accidentels à l'état initial.

Note

On peut utiliser la fonction rand()au lieu de la fonction MathRand().

© 2000-2024, MetaQuotes Ltd.

MathRound

Rend la valeur arrondie jusqu'au nombre entier plus proche de la valeur numérique indiquée.

```
double  MathRound(
   double  value      // valeur arrondit
   );
```

Paramètres

value

[in] La valeur numérique pour l'arrondissement.

La valeur rendue

La valeur arrondie jusqu'à le nombre entier le plus proche.

Note

On peut utiliser la fonction round() au lieu de la fonction MathRound().

© 2000-2024, MetaQuotes Ltd.

MathSin

Rend le sinus de l'angle indiqué.

```
double  MathSin(
   double  value      // argument dans les radians
   );
```

Paramètres

value

[in] L'angle dans les radians.

La valeur rendue

Le sinus de l'angle, indiqués dans les radians. Rend la valeur dans le domaine de -1 jusqu'au 1.

Note

On peut utiliser la fonction sin()au lieu de la fonction MathSin().

© 2000-2024, MetaQuotes Ltd.

MathSqrt

Rend la racine carrée du nombre.

```
double  MathSqrt(
   double  value      // nombre positif
   );
```

Paramètres

value

[in] La valeur positive numérique.

La valeur rendue

La racine carrée du nombre value. Si value est négative, MathSqrt rend NaN (valeur indéterminée).

Note

On peut utiliser la fonction sqrt()au lieu de la fonction MathSqrt().

Voir aussi

Les types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathSrand

Établit l'état initial de la génération des nombres entiers quasi-aléatoires.

```
void  MathSrand(
   int  seed      // nombre initialisant
   );
```

Paramètres

seed

[in] Le nombre initial pour une série de nombres accidentels.

La valeur rendue

Il n'y a pas de valeur rendue.

Note

La fonction MathRand() est destinée à la génération de la succession des nombres quasi-aléatoires. L' appel MathSrand() avec le nombre défini initialisant permet de recevoir toujours la même succession des nombres quasi-aléatoires.

Pour la réception garantie de la succession qui ne se répéte pas utilisez l'appel MathSrand(GetTickCount()), puisque la valeurGetTickCount()augmente dès le moment du lancement du système d'exploitation et ne se répète pas pendant 49 jours, jusqu'au le moment quand le compteur inséré des millisecondes ne débordera pas. L'utilisation MathSrand(TimeCurrent()) ne convient pas patce que la fonctionTimeCurrent()rend le temps de l'arrivée du dernier tick, qui ne peut ne pas être changé longtemps, par exemple, aux jours fériés.

Il est mieux de faire l'initialisation du générateur des nombres quasi-aléatoires à l'aide de MathSrand () pour les indicateurs et les experts dans le gestionnaire OnInit (), cela délivrera des lancements suivants multiples du générateur dans OnTick() et OnCalculate().

Au lieu de la fonction MathSrand() on peut utiliser la fonctionsrand().

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
#property description "L'indicateur présente le théorème central limite, qui déclare:'
#property description "La somme du nombre assez grand de variables aléatoires faibleme
#property description "ayant approximativement les mêmes échelles (ou nul opérande de
#property description "ne contribue pas le dépôt définissant à la somme), a la distrib

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- les propriétés de la construction graphique
#property indicator_label1  "Label"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRoyalBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  5
//--- variable input
input int      sample_number=10;
//--- le tampon d'indicateur pour le dessin de la distribution
double         LabelBuffer[];
//--- le compteur des ticks
double         ticks_counter;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- le liage du tableau et du tampon d'indicateur
   SetIndexBuffer(0,LabelBuffer,INDICATOR_DATA);
//--- déploierons le tampon d'indicateur du présent au passé
   ArraySetAsSeries(LabelBuffer,true);
//--- initialisons le générateur des nombres aléatoires
   MathSrand(GetTickCount());
//--- initialisons le compteur des ticks
   ticks_counter=0;
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le tampon d'indicateur sera remis à zéro au compteur nul
   if(ticks_counter==0) ArrayInitialize(LabelBuffer,0);
//--- augmentons le compteur
   ticks_counter++;
//--- il faut périodiquement enlever le compteur des ticks pour ranimer la distributio
   if(ticks_counter>100)
     {
      Print("on a remit à zéro les valeurs de l'indicateur, commencerons à remplir les
      ticks_counter=0;
     }
//--- recevrons un échantillon de valeurs aléatoires en tant que la somme de trois nor
   for(int i=0;i<sample_number;i++)
     {
```

```
      //---le calcul de l'index de la cellule, où tombera un nombre aléatoire comme la
      int rand_index=0;
      //--- recevrons trois nombres aléatoires de 0 jusqu'à 7
      for(int k=0;k<3;k++)
        {
         //--- le reste de la division sur 7 rendra la valeur de 0 jusqu'à 6
         rand_index+=MathRand()%7;
        }
      //--- augmentons sur une unité la valeur dans la cellule avec le numéro rand_inc
      LabelBuffer[rand_index]++;
     }
//--- la sortie du gestionnaire OnCalculate()
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

MathTan

Rend la tangent du nombre.

```
double  MathTan(
   double  rad      // argument dans les radians
   );
```

Paramètres

rad

[in] L'angle dans les radians.

La valeur rendue

La tangent du nombre rad. Si rad est plus ou égal au 263 ou moins que ou égal au -263, on perd la valeur et la fonction rend un nombre indéterminé.

Note

On peut utiliser la fonction tan()au lieu de la fonction MathTan().

Voir aussi

Les types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathIsValidNumber

Vérifie l'exactitude d'un nombre réel

```
bool  MathIsValidNumber(
   double  number      // nombre pour vérifier
   );
```

Paramètres

number

[in] Le nombre vérifié.

La valeur rendue

Rend true, si la valeur vérifiée est le nombre réel admissible. Si la valeur vérifiée est un plus ou le moins l'infinité, ou "pas un nombre" (NaN - not a number), la fonction rend false.

Exemple:

```
   double abnormal=MathArcsin(2.0);
   if(!MathIsValidNumber(abnormal)) Print("Attention! MathArcsin(2.0) = ",abnormal);
```

Voir aussi

Les types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathExpm1

Retourne la valeur de l'expression MathExp(x)-1.

```
double  MathExpm1(
   double  value      // puissance du nombre e
   );
```

Paramètres

value

[in] Le nombre spécifiant la puissance.

Valeur de Retour

Une valeur de type double. En cas d'overflow, la fonction retourne INF (infini), en cas d'underflow MathExpm1 retourne 0.

Note

Aux valeurs de *x* proches de 0, la fonction MathExpm1(x) génère des valeurs plus précises que la fonction MathExp(x)-1.

Au lieu d'utiliser la fonction MathExpm1(), vous pouvez utiliser la fonction expm1().

Voir aussi

Types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathLog1p

Retourne la valeur de l'expression MathLog(1+x).

```
double  MathLog1p(
   double  value      // valeur prenant le logarithme
   );
```

Paramètres

value

[in] La valeur dont le logarithme sera calculé.

Valeur de Retour

Le logarithme naturel de la valeur (valeur + 1) en cas de succès. Si value est < -1, la fonction retourne NaN (valeur indéfinie). Si value est égale à -1, la fonction retourne INF (infini).

Note

Aux valeurs de x proches de 0, la fonction MathLog1p(x) génère des valeurs plus précises que la fonction MathLog(x)-1.

Au lieu d'utiliser la fonction MathLog1p(), vous pouvez utiliser la fonction log1p().

Voir aussi

Types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathArccosh

Retourne l'arc cosinus hyperbolique.

```
double  MathArccosh(
   double  value     // 1 <= valeur < ∞
   );
```

Paramètres

value

[in] La valeur pour laquelle l'arc cosinus hyperbolique est calculé.

Valeur de Retour

L'arc cosinus hyperbolique du nombre. Si value est inférieur à +1, la fonction retourne NaN (valeur indéfinie).

Note

Au lieu d'utiliser la fonction MathArccosh(), vous pouvez utiliser la fonction acosh().

Voir aussi

Types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathArcsinh

Retourne l'arc sinus hyperbolique.

```
double  MathArcsinh(
   double  value      // -∞ < valeur < +∞
   );
```

Paramètres

val

[in] La valeur pour laquelle l'arc sinus hyperbolique est calculé.

Valeur de Retour

L'arc sinus hyperbolique du nombre.

Note

Au lieu d'utiliser la fonction MathArcsinh(), vous pouvez utiliser la fonction asinh().

Voir aussi

Types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathArctanh

Retourne l'arc tangente hyperbolique.

```
double  MathArctanh(
   double  value      // valeur dans l'intervalle -1 < valeur < 1
   );
```

Paramètres

value

[in] Nombre dans l'intervalle -1 < valeur < 1, qui représente la tangente.

Valeur de Retour

L'arc tangente hyperbolique du nombre.

Note

Au lieu d'utiliser la fonction MathArctanh(), vous pouvez utiliser la fonction atanh().

© 2000-2024, MetaQuotes Ltd.

MathCosh

Retourne le cosinus hyperbolique du nombre.

```
double  MathCosh(
   double  value      // nombre
   );
```

Paramètres

value

[in] Valeur.

Valeur de Retour

Le cosinus hyperbolique du nombre, valeur dans l'intervalle +1 < infini positif.

Note

Au lieu d'utiliser la fonction MathCosh(), vous pouvez utiliser la fonction cosh().

© 2000-2024, MetaQuotes Ltd.

MathSinh

Retourne le sinus hyperbolique du nombre.

```
double  MathSinh(
   double  value      // nombre
   );
```

Paramètres

value

[in] Valeur.

Valeur de Retour

Le sinus hyperbolique du nombre.

Note

Au lieu d'utiliser la fonction MathSinh(), vous pouvez utiliser la fonction sinh().

© 2000-2024, MetaQuotes Ltd.

MathTanh

Retourne la tangente hyperbolique du nombre.

```
double  MathTanh(
   double  value      // nombre
   );
```

Paramètres

value

[in] Valeur.

Valeur de Retour

La tangente hyperbolique du nombre, valeur dans l'intervalle -1 à +1.

Note

Au lieu d'utiliser la fonction MathTanh(), vous pouvez utiliser la fonction tanh().

Voir aussi

Types réels (double, float)

© 2000-2024, MetaQuotes Ltd.

MathSwap

Change l'ordre des octets dans la valeur d'un type ushort.

```
ushort  MathSwap(
   ushort  value      // valeur
   );
```

Paramètres

value

[in] Valeur pour changer l'ordre des octets.

Valeur de Retour

la valeur ushort avec l'ordre inverse des octets.

MathSwap

Change l'ordre des octets dans la valeur d'un type uint.

```
uint  MathSwap(
   uint   value      // valeur
   );
```

Paramètres

value

[in] Valeur pour changer l'ordre des octets.

Valeur de Retour

la valeur uint avec l'ordre inverse des octets.

MathSwap

Change l'ordre des octets dans la valeur d'un type ulong.

```
ulong  MathSwap(
   ulong  value      // valeur
   );
```

Paramètres

value

[in] Valeur pour changer l'ordre des octets.

Valeur de Retour

la valeur ulong avec l'ordre inverse des octets.

Voir également

Fonctions réseau, SocketRead, SocketSend, SocketTlsRead, SocketTlsReadAvailable, SocketTlsSend

© 2000-2024, MetaQuotes Ltd.

Fonctions de chaîne

Le groupe des fonctions destinées au travail avec les données du type string.

Fonction	Action
StringAdd	Ajoute la chaîne selon la place indiquée par la sous-chaîne
StringBufferLen	Rend la grandeur du buffer distribué pour la chaîne.
StringCompare	Compare deux chaînes et rend 1 si la première chaîne est plus que deuxième; 0 - si les chaînes sont égales; -1 (moins un) - si la première chaîne est moins que la deuxième
StringConcatenate	Forme la chaîne des paramètres transmis
StringFill	Remplit la chaîne indiquée selon la place par les symboles indiqués
StringFind	La recherche de la sous-chaîne dans la chaîne
StringGetCharacter	Rend la valeur du symbole disposé dans la position indiquée de la chaîne
StringInit	Initialise la chaîne par les symboles indiqués et assure la grandeur indiquée de la chaîne
StringLen	Rend le nombre de symboles dans la chaîne
StringSetLength	Définit une longueur (en caractères) pour une chaîne de caractères
StringReplace	Remplace tous les sous-chaînes recherchés dans la chaîne pour la séquence spécifiée des caractères.
StringReserve	Réserve un buffer de la taille spécifiée pour une chaîne de caractères en mémoire
StringSetCharacter	Rend la copie de la chaîne avec la valeur changée du symbole dans la position indiquée
StringSplit	Reçoit les sous-chaînes de la chaîne indiquée par un séparateur spécifié et rend le montant des sous-chaînes reçus
StringSubstr	Extrait la sous-chaîne de la chaîne de texte qui commence de la position indiquée
StringToLower	Transforme tous les symboles de la chaîne indiquée aux symboles minuscules selon la place
StringToUpper	Transforme tous les symboles de la chaîne indiquée aux symboles majuscules selon la place
StringTrimLeft	Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation du début de la chaîne
StringTrimRight	Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation à la fin de la chaîne

© 2000-2024, MetaQuotes Ltd.

StringAdd

Ajoute la chaîne selon la place indiquée par la sous-chaîne.

```
bool  StringAdd(
   string&  string_var,        // chaîne, à qui est ajouté
   string   add_substring      // chaîne ajoutée
   );
```

Paramètres

string_var

[in][out] La chaîne, qui sera complétée.

add_substring

[in] La chaîne, qui sera ajoutée à la fin de la ligne initiale.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false.Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Exemple:

```
void OnStart()
  {
   long length=1000000;
   string a="a",b="b",c;
//--- premier moyen
   uint start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);

//--- deuxième moyen
   start=GetTickCount();
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   stop=GetTickCount();
   Print("time for 'StringAdd(a,b)' = ",(stop-start)," milliseconds, i = ",i);

//--- troisième moyen
   start=GetTickCount();
   a="a"; // initialisons de nouveau la variable a
   for(i=0;i<length;i++)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       StringConcatenate(c,a,b);
      }
    stop=GetTickCount();
    Print("time for 'StringConcatenate(c,a,b)' = ",(stop-start)," milliseconds, i = ",i
  }
```

Voir aussi

StringConcatenate, StringSplit, StringSubstr

© 2000-2024, MetaQuotes Ltd.

StringBufferLen

Rend la grandeur du buffer distribué pour la chaîne.

```
int  StringBufferLen(
   string  string_var      // chaîne
   )
```

Paramètres

string_var

[in] La chaîne.

La valeur rendue

La valeur 0 signifie que la chaîne est constante et c'est interdit le contenu du buffer. -1 signifie que la chaîne appartient au terminal de client et le changement du contenu du buffer peut amener aux résultats incertains.

Exemple:

```
void OnStart()
  {
   long length=1000;
   string a="a",b="b";
//---
   long i;
   Print("before: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   Print("after: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
  }
```

Voir aussi

StringAdd, StringInit, StringLen, StringFill

© 2000-2024, MetaQuotes Ltd.

StringCompare

Compare deux chaînes et retourne le résultat de la comparaison comme un entier.

```
int  StringCompare(
   const string&  string1,                // une première chaîne à comparer
   const string&  string2,                // une deuxième chaîne à comparer
   bool           case_sensitive=true     // le mode du compte du registre des lettre
   );
```

Paramètres

string1

[in] Une première chaîne.

string2

[in] Une deuxième chaîne.

case_sensitive=true

[in] Le mode du compte du registre des lettres. Si la valeur est true, c'est "A">"a". Si la valeur est false, c'est "A"="a". Par défaut la valeur du paramètre est true.

La valeur rendue

- -1 (moins un), si string1<string2
- 0 (le zéro), si string1=string2
- 1 (un), si string1>string2

Note

Les chaînes sont comparées caractère par caractère, les caractères sont comparés en ordre alphabétique avec la page de code actuelle.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- qu'est-ce que est plus grande, la pomme ou la maison?
   string s1="Apple";
   string s2="home";

//--- comparerons en tenant compte du registre
   int result1=StringCompare(s1,s2);
   if(result1>0) PrintFormat("la comparaison en tenant compte du registre: %s > %s",s1
   else
     {
      if(result1<0)PrintFormat("La comparaison en tenant compte du registre: %s < %s",
      else PrintFormat("La comparaison en tenant compte du registre: %s = %s",s1,s2);
     }

//--- comparerons sans tenant compte du registre
   int result2=StringCompare(s1,s2,false);
   if(result2>0) PrintFormat("La comparaison sans tenant compte du registre: %s > %s",
   else
     {
      if(result2<0)PrintFormat("La comparaison sans tenant compte du registre: %s < %s
      else PrintFormat("La comparaison sans tenant compte du registre: %s = %s",s1,s2)
     }
/* Résultat:
    La comparaison en tenant compte du registre: Apple < home
    La comparaison sans tenant compte du registre: Apple < home
*/
  }
```

Voir aussi

Type string, CharToString(), ShortToString(), StringToCharArray(), StringToShortArray(), StringGetCharacter(), Utilisation de la page de code

© 2000-2024, MetaQuotes Ltd.

StringConcatenate

Forme la chaîne des paramètres transmis et rend la grandeur de la chaîne formée. Les paramètres peuvent avoir n'importe quel type. Le nombre de paramètres ne peut pas être moins 2 et ne peut pas excéder 64.

```
int  StringConcatenate(
   string&  string_var,    // chaîne pour la formation
   void argument1          // premier paramètre de n'importe quel type simple
   void argument2          // deuxième paramètre de n'importe quel type simple
   ...                     // paramètre suivant de n'importe quel type simple
   );
```

Paramètres

string_var

[out] La chaîne qui sera formée à la suite de la concaténation.

argumentN

[in] N'importe quelles valeurs divisées par les virgules.De 2 à 63 paramètres de n'importe quel type simple.

La valeur rendue

Rend la longueur de chaîne, formée par la concaténation de paramètres transformés dans le type string. Les paramètres seront transformés en chaînes selon les mêmes règles que dans les fonctions Print() et Comment().

Voir aussi

StringAdd, StringSplit, StringSubstr

© 2000-2024, MetaQuotes Ltd.

StringFill

Remplit la chaîne indiquée selon la place par les symboles indiqués.

```
bool  StringFill(
   string&   string_var,        // chaîne pour le remplissage
   ushort    character          // symbole, par lequel on remplit la chaîne
   );
```

Paramètres

string_var

[in][out] La chaîne, qui sera remplie par le symbole indiqué.

character

[in] Le symbole, par qui on remplira la chaîne.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false. Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Note

Le remplissage de la chaîne selon la place signifie que les symboles sont insérés directement dans la chaîne sans opérations intermédiaires de la création de la nouvelle chaîne et le copiage. Cela permet de diminuer le temps de travail avec la chaîne dans cette fonction.

Exemple:

```
void OnStart()
  {
   string str;
   StringInit(str,20,'_');
   Print("str = ",str);
   StringFill(str,0);
   Print("str = ",str,": StringBufferLen(str) = ", StringBufferLen(str));
  }
// Résultat
//   str = ____________________
//   str =  : StringBufferLen(str) = 20
//
```

Voir aussi

StringBufferLen, StringLen, StringInit

© 2000-2024, MetaQuotes Ltd.

StringFind

La recherche de la sous-chaîne dans la chaîne.

```
int  StringFind(
   string  string_value,        // chaîne, où on fait la recherche
   string  match_substring,     // ce qui est recherché
   int     start_pos=0          // par quelle position commencer la recherche
   );
```

Paramètres

string_value

[in] La chaîne, dans laquelle la recherche est faite.

match_substring

[in] La sous-chaîne recherchée.

start_pos=0

[in] La position dans la chaîne, à partir de laquelle on doit commencer la recherche.

La valeur rendue

Rend le numéro de la position dans la chaîne, par laquelle la sous-chaîne recherchée commence, ou -1, si la sous-chaîne n'est pas trouvée.

Voir aussi

StringSubstr, StringGetCharacter, StringLen, StringLen

© 2000-2024, MetaQuotes Ltd.

StringGetCharacter

Rend la copie de la chaîne avec la valeur changée du symbole dans la position indiquée.

```
ushort  StringGetCharacter(
   string  string_value,      // chaîne
   int     pos                // position du symbole dans la chaîne
   );
```

Paramètres

string_value

[in] La chaîne.

pos

[in] La position du symbole dans la chaîne. Peut être de 0 jusqu'à StringLen(text) -1.

La valeur rendue

Le code du symbole ou 0 en cas d'une erreur. Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Voir aussi

StringSetCharacter, StringBufferLen, StringLen, StringFill, StringInit, StringToCharArray, StringToShortArray

© 2000-2024, MetaQuotes Ltd.

StringInit

Initialise la chaîne par les symboles indiqués et assure la grandeur indiquée de la chaîne.

```
bool  StringInit(
   string&   string_var,        // chaîne pour l'initialisation
   int       new_len=0,         // longueur demandée de la chaîne après l'initialisatio
   ushort    character=0        // symbole par lequel la chaîne sera rempli
   );
```

Paramètres

string_var

[in][out] La chaîne, qui doit être initialisée ou déinitialisé.

new_len=0

[in] La longueur de la chaîne après l'initialisation. Si la longueur est=0, elle déinitialise la chaîne, c'est-à-dire, le buffer de la chaîne est dégagé et l'adresse du buffer se met au zéro.

character=0

[in] Le symbole pour le remplissage de la chaîne.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false. Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Note

Si character=0 et la grandeur new_len>0, le buffer de la chaîne de la longueur indiquée sera distribué et sera rempli par les zéros. La longueur de la chaîne sera égale au zéro, parce que tout le buffer sera rempli par les terminateurs de la chaîne.

Exemple:

```
void OnStart()
  {
//---
   string str;
   StringInit(str,200,0);
   Print("str = ",str,": StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/*  Résultat
str = : StringBufferLen(str) = 200   StringLen(str) = 0
*/
```

Voir aussi

StringBufferLen, StringLen

© 2000-2024, MetaQuotes Ltd.

StringLen

Rend le nombre de symboles dans la chaîne.

```
int  StringLen(
   string  string_value      // chaîne
   );
```

Paramètres

string_value

[in] La chaîne pour le calcul de la longueur.

La valeur rendue

Le nombre de symboles dans la chaîne sans le zéro terminant.

Voir aussi

StringBufferLen, StringTrimLeft, StringTrimRight, StringToCharArray, StringToShortArray

© 2000-2024, MetaQuotes Ltd.

StringSetLength

Définit une longueur (en caractères) pour une chaîne de caractères.

```
bool  StringSetLength(
   string&    string_var,       // chaîne de caractères
   uint       new_length        // nouvelle longueur de la chaîne de caractères
   );
```

Parameters

string_var

[in][out] Chaîne de caractères pour laquelle la nouvelle longueur est à définir.

new_capacity

[in] Longueur désirée de la chaîne en nombre de caractères. Si new_length est inférieur à la taille actuelle, les caractères en excès sont ignorés.

Valeur de Retour

En cas d'exécution avec succès, retourne true, sinon false. Pour obtenir le code d'erreur, il faut appeler la fonction GetLastError().

Note

La fonction StringSetLength() ne change pas la taille du buffer alloué pour une chaîne de caractères.

Voir également

StringLen, StringBufferLen, StringReserve StringInit, StringSetCharacter

© 2000-2024, MetaQuotes Ltd.

StringReplace

Remplace tous les sous-chaînes recherchés dans la chaîne pour la séquence spécifiée des caractères.

```
int  StringReplace(
   string&         str,              // la chaîne, où sera faite le remplacement
   const string    find,             // la sous-chaîne recherchée
   const string    replacement       // la sous-chaîne, qui sera insérée dans les plac
   );
```

Paramètres

str

[in][out] La chaîne dans laquelle il faut faire le changement.

find

[in] La sous-chaîne recherchée pour le remplacement.

replacement

[in] La chaîne qui doit être insérée à la place de la chaîne trouvée.

La valeur rendue

Le nombre de remplacements produits en cas du succès, en cas de l'erreur c'est -1. Pour recevoir le codede l'erreur il faut appeler la fonction GetLastError().

Note

Si la fonction a fonctionné avec succès, mais les remplacements ne sont pas produits (la sous-chaîne remplacée n'est pas trouvée), 0 revient

La cause de l'erreur peuvent être des paramètres incorrects *str* ou *find* (la chaîne vide ou n'est pas initialisée, à voir StringInit()). En outre l'erreur apparaîtra, s'il ne suffit pas la mémoire pour l'achèvement des remplacements.

Exemple:

```
  string text="The quick brown fox jumped over the lazy dog.";
  int replaced=StringReplace(text,"quick","slow");
  replaced+=StringReplace(text,"brown","black");
  replaced+=StringReplace(text,"fox","bear");
  Print("Replaced: ", replaced,". Result=",text);

//  Le résultat
//  Replaced: 3. Result=The slow black bear jumped over the lazy dog.
//
```

Voir aussi

StringSetCharacter(), StringSubstr()

© 2000-2024, MetaQuotes Ltd.

StringReserve

Réserve un buffer de la taille spécifiée pour une chaîne de caractères en mémoire.

```
bool  StringReserve(
   string&     string_var,        // chaîne de caractères
   uint        new_capacity       // taille du buffer pour stocker une chaîne de caracté
   );
```

Parameters

string_var

[in][out] Chaîne de caractères pour laquelle la taille est à changer.

new_capacity

[in] Taille du buffer souhaitée. Si la taille new_capacity est inférieure à la longueur de la chaîne, la taille du buffer actuel n'est pas changée.

Valeur de Retour

En cas d'exécution avec succès, retourne true, sinon false. Pour obtenir le code d'erreur, il faut appeler la fonction GetLastError().

Note

De façon générale, la taille de la chaîne de caractères n'est pas égale à la taille du buffer prévu pour stocker la chaîne. Lors de la création d'une chaîne, le buffer correspondant est habituellement alloué avec une marge. La fonction StringReserve() permet de gérer la taille du buffer et de spécifier la taille optimale pour des opérations futures.

Contrairement à StringInit(), la fonction StringReserve() ne change pas le contenu de la chaîne de caractères et ne la remplit pas avec des caractères.

Exemple:

```
void OnStart()
  {
   string s;
//--- vérifie la vitesse d'opération sans utiliser StringReserve
   ulong t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_no_reserve=GetMicrosecondCount()-t0;
   s=NULL;
//--- effectuons la même opération avec StringReserve
   StringReserve(s,1024 * 3);
   t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_reserve=GetMicrosecondCount()-t0;
//--- vérification du temps
   Print("Test avec StringReserve effectué en "+(string)msc_reserve+" ms");
   Print("Test sans StringReserve effectué en "+(string)msc_no_reserve+" ms");
```

© 2000-2024, MetaQuotes Ltd.

```
/* Résultat :
    Test avec StringReserve effectué en 50 ms
    Test sans StringReserve effectué en 121 ms
*/
  }
```

Voir également

StringBufferLen, StringSetLength, StringInit, StringSetCharacter

© 2000-2024, MetaQuotes Ltd.

StringSetCharacter

Rend la copie de la chaîne avec la valeur changée du symbole dans la position indiquée.

```
bool  StringSetCharacter(
   string&   string_var,        // chaîne
   int       pos,               // position
   ushort    character          // symbole
   );
```

Paramètres

string_var

[in][out] La chaîne.

pos

[in] La position du symbole dans la chaîne. Peut être de 0 jusqu'à StringLen(text).

character

[in] Le code de symbole Unicode.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false.Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Note

Si la valeur pos est moins que la longueur de la chaîne et la valeur du code de symbole est= 0, la chaîne est coupée (mais la grandeur du buffer, distribuée pour la chaîne reste invariable). La longueur de la chaîne devient égale à la signification pos.

Si la signification du paramètre *pos* est égale à la signification de la longueur de la chaîne, le symbole indiqué est ajouté à la fin de la chaîne,et ainsi la longueur de la chaîne augmente sur une unité.

Exemple:

```
void OnStart()
  {
   string str="0123456789";
   Print("before: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- ajoutez la valeur zéro au milieu de la chaîne
   StringSetCharacter(str,6,0);
   Print("  after: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- ajoutons le symbole à la fin de la chaîne
   int size=StringLen(str);
   StringSetCharacter(str,size,'+');
   Print("addition: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
```

© 2000-2024, MetaQuotes Ltd.

```
/* Résultat
    before: str = 0123456789 ,StringBufferLen(str) = 0   StringLen(str) = 10
     after: str = 012345 ,StringBufferLen(str) = 16   StringLen(str) = 6
  addition: str = 012345+ ,StringBufferLen(str) = 16   StringLen(str) = 7
*/
```

Voir aussi

StringBufferLen, StringLen, StringFill, StringInit, CharToString, ShortToString, CharArrayToString, ShortArrayToString

© 2000-2024, MetaQuotes Ltd.

StringSplit

Reçoit les sous-chaînes selon un séparateur spécifié de la chaîne indiquée et rend le nombre de sous-chaînes reçus.

```
int  StringSplit(
   const string   string_value,       // la chaîne pour la recherche des sous-chaînes
   const ushort   separator,          // le séparateur, à l'aide duquel les sous-chaîr
   string         & result[]          // le tableau transmis selon le lien pour la réc
   );
```

Paramètres

string_value

[in] La chaîne d'où il faut recevoir les sous-chaînes. La chaîne elle-même ne change pas.

pos

[in] Le code du symbole du séparateur. Pour recevoir le code, vous pouvez utiliser la fonction StringGetCharacter().

result[]

[out] Le tableau des chaînes, où se placent les sous-chaînes reçus.

La valeur rendue

Le nombre de chaînes réçues dans le tableau result[]. Si le séparateur dans la chaîne transmise n'est pas trouvée, alors seulement une chaîne initiale sera placée au tableau.

Si la chaîne string_value est vide ou NULL, la fonction rendra le zéro. Dans le cas de l'erreur, la fonction rendra -1. Pour recevoir le codede l'erreur il faut appeler la fonction GetLastError().

Exemple:

```
string to_split="_maman_a_lavé_la_fenêtre_"; // la chaîne pour se diviser en sous-chaî
   string sep="_";                           // le séparateur comme le symbole
   ushort u_sep;                             // le code du symbole du séparateur
   string result[];                          // le tableau pour recevoir les chaînes
   //--- recevrons le code du séparateur
   u_sep=StringGetCharacter(sep,0);
   //--- divisons la chaîne en sous-chaînes
   int k=StringSplit(to_split,u_sep,result);
   //--- déduisons le commentaire
   PrintFormat("On a reçu le nombre de chaînes: %d. On a utilisé le séparateur '%s' av
   //--- maintenant nous déduisons toutes les chaînes reçues
   if(k>0)
     {
      for(int i=0;i<k;i++)
        {
         PrintFormat("result[%d]=\"%s\"",i,result[i]);
        }
     }
```

Voir aussi

StringReplace(), StringSubstr(), StringConcatenate()

© 2000-2024, MetaQuotes Ltd.

StringSubstr

Extrait une sous-chaîne d'une chaîne de texte à partir de la position spécifiée..

```
string  StringSubstr(
   string  string_value,      // chaîne
   int     start_pos,         // par quelle position commencer
   int     length=-1          // longueur de la chaîne extraite
   );
```

Paramètres

string_value

[in] La chaîne, d'où doit être extrait la sous-chaîne.

start_pos

[in] La position initiale de la sous-chaîne. Peut - être de 0 jusqu'à StringLen(text) -1.

length=-1

[in] La longueur de la sous-chaîne extraite. Si la valeur du paramètre est égale au -1 ou le paramètre n'est pas spécifié, la sous-chaîne sera extrait, à partir de la position indiquée et jusqu'à la fin de la chaîne.

La valeur rendue

La copie du sous-chaîne extrait s'il est possible, autrement revient la chaîne vide.

Voir aussi

StringSplit, StringFind, StringGetCharacter

© 2000-2024, MetaQuotes Ltd.

StringToLower

Transforme tous les symboles de la chaîne indiquée aux symboles minuscules selon la place.

```
bool  StringToLower(
   string&  string_var      // chaîne à traiter
   );
```

Paramètres

string_var

[in][out] La chaîne.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false.Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Voir aussi

StringToUpper, StringTrimLeft, StringTrimRight

© 2000-2024, MetaQuotes Ltd.

StringToUpper

Transforme tous les symboles de la chaîne indiquée aux symboles majuscules selon la place.

```
bool  StringToUpper(
   string&  string_var      // chaîne à traiter
   );
```

Paramètres

string_var

[in][out] La chaîne.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false.Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Voir aussi

StringToLower, StringTrimLeft, StringTrimRight

© 2000-2024, MetaQuotes Ltd.

StringTrimLeft

Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation du début de la chaîne jusqu'au premier symbole significatif. La chaîne est modifiée selon la place

```
int  StringTrimLeft(
   string&  string_var      // chaîne à couper
   );
```

Paramètres

string_var

[in][out] La chaîne, qui sera coupé à partir de la gauche.

La valeur rendue

Rend le nombre de symboles coupés.

Voir aussi

StringTrimRight, StringToLower, StringToUpper

© 2000-2024, MetaQuotes Ltd.

StringTrimRight

Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation à la fin de la chaîne. La chaîne est modifiée selon la place.

```
int  StringTrimRight(
   string&  string_var      // chaîne à couper
   );
```

Paramètres

string_var

[in][out] La chaîne, qui sera coupé à droite.

La valeur rendue

Rend le nombre de symboles coupés.

Voir aussi

StringTrimLeft, StringToLower, StringToUpper

© 2000-2024, MetaQuotes Ltd.

La date et le temps

Le groupe des fonctions assurant le travail avec les données du type datetime (un nombre entier qui représente le nombre de secondes, passées dès 0 heures du 1 janvier 1970).

Pour organiser les compteurs et la minuterie à haute résolution il faut utiliser la fonction GetTickCount () qui donne la valeur en millisecondes.

Fonction	Action
TimeCurrent	Rend le dernier temps connu du serveur (le temps de l'arrivée de la dernière cotation) dans le format datetime
TimeTradeServer	Rend le temps calculé courant du serveur commercial
TimeLocal	Rend le temps local informatique dans le format datetime
TimeGMT	Rend le temps GMT au format datetime en tenant compte du passage sur l'heure d'hiver ou d'été par le temps local de l'ordinateur, sur lequel le terminal de client a été lancé
TimeDaylightSavings	Rend le signe du passage sur l'heure d'été/l'hiver
TimeGMTOffset	Rend la différence actuelle entre le temps GMT et le temps local de l'ordinateur en secondes en tenant compte du passage sur l'heure d'hiver ou d'été
TimeToStruct	Produit la conversion de la valeur du type datetime à la variable du type de la structure MqlDateTime
StructToTime	Produit la conversion de la valeur du type de la structure MqlDateTime à la variable du type datetime

© 2000-2024, MetaQuotes Ltd.

TimeCurrent

Rend le dernier temps connu du serveur, le temps de l'arrivée de la dernière cotation pour un des symboles choisis dans "L'aperçu du marché". Au gestionnaire OnTick() cette fonction rendra le temps du tick traité venant. Dans les autres cas (par exemple, l'appel dans les gestionnaires OnInit(), OnDeinit(), OnTimer() etc) c'est – le temps de l'arrivée de la dernière cotation selon n'importe quel symbole, accessible dans la fenêtre "L'aperçu du marché", ce temps, qui est montré dans le titre de cette fenêtre. La valeur du temps est formée sur un serveur commercial et ne dépend pas des des ajustements du temps sur l'ordinateur de l'utilisateur. Il y a 2 variantes de la fonction.

L'appel sans paramètres

```
datetime  TimeCurrent();
```

L'appel avec le paramètre du type MqlDateTime

```
datetime  TimeCurrent(
   MqlDateTime&  dt_struct      // variable du type de la structure
   );
```

Paramètres

dt_struct

[out] La variable du type de la structure MqlDateTime.

La valeur rendue

La valeur du type datetime

Note

Si la variable du type de la structure MqlDateTime a été transmise comme paramètre, elle s'est remplie de façon appropriée.

Pour organiser les compteurs et la minuterie à haute résolution il faut utiliser la fonction GetTickCount () qui donne la valeur en millisecondes.

При работе в тестере стратегий время последней котировки TimeCurrent() моделируется в соответствии с историческими данными.

© 2000-2024, MetaQuotes Ltd.

TimeTradeServer

Rend le temps calculé courant du serveur commercial. A la différence de la fonction TimeCurrent(), le calcul de la valeur du temps est produit dans le terminal de client et dépend des réglages du temps de l'ordinateur de l'utilisateur. Il y a 2 variantes de la fonction.

L'appel sans paramètres

```
datetime  TimeTradeServer();
```

L'appel avec le paramètre du type MqlDateTime

```
datetime  TimeTradeServer(
   MqlDateTime&  dt_struct      // variable du type de la structure
   );
```

Paramètres

dt_struct

[out] La variable du type de la structure MqlDateTime.

La valeur rendue

La valeur du type datetime

Note

Si la variable du type de la structure MqlDateTime a été transmise comme paramètre, elle s'est remplie de façon appropriée.

Pour organiser les compteurs et la minuterie à haute résolution il faut utiliser la fonction GetTickCount () qui donne la valeur en millisecondes.

При работе в тестере стратегий TimeTradeServer() всегда равно моделируемому серверному времени TimeCurrent().

© 2000-2024, MetaQuotes Ltd.

TimeLocal

Rend le temps local de l'ordinateur, sur lequel on a lancé le terminal de client. Il y a 2 variantes de la fonction.

L'appel sans paramètres

```
datetime  TimeLocal();
```

L'appel avec le paramètre du type MqlDateTime

```
datetime  TimeLocal(
   MqlDateTime&  dt_struct      // variable du type de la structure
   );
```

Paramètres

dt_struct

[out] La variable du type de la structure MqlDateTime.

La valeur rendue

La valeur du type datetime

Note

Si la variable du type de la structure MqlDateTime a été transmise comme paramètre, elle s'est remplie de façon appropriée.

Pour organiser les compteurs et la minuterie à haute résolution il faut utiliser la fonction GetTickCount () qui donne la valeur en millisecondes.

При работе в тестере стратегий локальное время TimeLocal() всегда равно моделируемому серверному времени TimeCurrent().

© 2000-2024, MetaQuotes Ltd.

TimeGMT

Rend le temps GMT au format datetime en tenant compte du passage sur l'heure d'hiver ou d'été par le temps local de l'ordinateur, sur lequel le terminal de client a été lancé. Il y a 2 variantes de la fonction.

L'appel sans paramètres

```
datetime  TimeGMT();
```

L'appel avec le paramètre du type MqlDateTime

```
datetime  TimeGMT(
   MqlDateTime&  dt_struct      // variable du type de la structure
   );
```

Paramètres

dt_struct

[out] La variable du type de la structure MqlDateTime.

La valeur rendue

La valeur du type datetime

Note

Si la variable du type de la structure MqlDateTime a été transmise comme paramètre, elle s'est remplie de façon appropriée.

Pour organiser les compteurs et la minuterie à haute résolution il faut utiliser la fonction GetTickCount () qui donne la valeur en millisecondes.

При работе в тестере стратегий время TimeGMT() всегда равно моделируемому серверному времени TimeTradeServer().

© 2000-2024, MetaQuotes Ltd.

TimeDaylightSavings

Rend la correction sur l'heure d'été en des secondes, si on a fait le passage à l'heure d'été. Dépend des réglages du temps de l'ordinateur de l'utilisateur.

```
int  TimeDaylightSavings();
```

La valeur rendue

Si on a fait le passage à l'heure d'hiver (standard), revient 0.

© 2000-2024, MetaQuotes Ltd.

TimeGMTOffset

Rend la différence actuelle entre le temps GMT et le temps local de l'ordinateur en secondes en tenant compte du passage sur l'heure d'hiver ou d'été. Dépend des réglages du temps de l'ordinateur de l'utilisateur.

```
int  TimeGMTOffset();
```

La valeur rendue

La valeur du type int, qui présente la différence courante entre le temps GMT et le temps local de l'ordinateur TimeLocal() en secondes.

```
TimeGMTOffset() =  TimeGMT() - TimeLocal()
```

© 2000-2024, MetaQuotes Ltd.

TimeToStruct

Produit la conversion de la valeur du type datetime (le nombre de secondes depuis le 01.01.1970) à la variable du type de la structure MqlDateTime.

```
bool  TimeToStruct(
   datetime      dt,            // date et temps
   MqlDateTime&  dt_struct      // structure pour l'adoption de valeurs
   );
```

Paramètres

dt

[in] La valeur de la date pour la conversion.

dt_struct

[out] La variable du type de la structure MqlDateTime.

La valeur rendue

true - en cas du succès, autrement false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

StructToTime

Produit la conversion de la valeur du type de la structure MqlDateTime à la variable du type datetime et rend la valeur reçue.

```
datetime  StructToTime(
   MqlDateTime&  dt_struct      // structure de la date et du temps
   );
```

Paramètres

dt_struct

[in] La variable du type de la structure MqlDateTime.

La valeur rendue

La valeur du type datetime, qui contient le nombre de secondes depuis le 01.01.1970.

© 2000-2024, MetaQuotes Ltd.

L'information sur le compte

Les fonctions qui rendent des paramètres du compte courant.

Fonction	Action
AccountInfoDouble	Rend la valeur du type double de la propriété correspondante du compte
AccountInfoInteger	Rend la valeur du type entier (bool,int ou long) de la propriété correspondante du compte
AccountInfoString	Rend la valeur du type string de la propriété correspondante du compte

© 2000-2024, MetaQuotes Ltd.

AccountInfoDouble

Rend la valeur de la propriété correspondante du compte.

```
double  AccountInfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. La valeur peut être une des valeurs ENUM_ACCOUNT_INFO_DOUBLE.

La valeur rendue

La valeur du type double.

Exemple:

```
void OnStart()
  {
//--- montrons toute l'information disponible de la fonction AccountInfoDouble()
   printf("ACCOUNT_BALANCE =  %G",AccountInfoDouble(ACCOUNT_BALANCE));
   printf("ACCOUNT_CREDIT =  %G",AccountInfoDouble(ACCOUNT_CREDIT));
   printf("ACCOUNT_PROFIT =  %G",AccountInfoDouble(ACCOUNT_PROFIT));
   printf("ACCOUNT_EQUITY =  %G",AccountInfoDouble(ACCOUNT_EQUITY));
   printf("ACCOUNT_MARGIN =  %G",AccountInfoDouble(ACCOUNT_MARGIN));
   printf("ACCOUNT_MARGIN_FREE =  %G",AccountInfoDouble(ACCOUNT_MARGIN_FREE));
   printf("ACCOUNT_MARGIN_LEVEL =  %G",AccountInfoDouble(ACCOUNT_MARGIN_LEVEL));
   printf("ACCOUNT_MARGIN_SO_CALL = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL));
   printf("ACCOUNT_MARGIN_SO_SO = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_SO));
  }
```

Voir aussi

SymbolInfoDouble, SymbolInfoString, SymbolInfoInteger, PrintFormat

© 2000-2024, MetaQuotes Ltd.

AccountInfoInteger

Rend la valeur de la propriété correspondante du compte.

```
long  AccountInfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. La valeur peut être une des valeurs ENUM_ACCOUNT_INFO_INTEGER.

La valeur rendue

La valeur du type long.

Note

La propriété doit être un des types bool, int ou long.

Exemple:

```
void OnStart()
  {
//--- montrons toute l'information accessible de la fonction AccountInfoInteger()
   printf("ACCOUNT_LOGIN =  %d",AccountInfoInteger(ACCOUNT_LOGIN));
   printf("ACCOUNT_LEVERAGE =  %d",AccountInfoInteger(ACCOUNT_LEVERAGE));
   bool thisAccountTradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_ALLOWED);
   bool EATradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_EXPERT);
   ENUM_ACCOUNT_TRADE_MODE tradeMode=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(ACCOU
   ENUM_ACCOUNT_STOPOUT_MODE stopOutMode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoIntegei

//--- informons de la possibilité d'exécuter les opérations commerciales
   if(thisAccountTradeAllowed)
      Print("Le commerce pour ce compte est permis");
   else
      Print("Le commerce pour ce compte est interdit!");

//---  éclaircissons - s'il est possible de faire du commerce sur ce compte par les ex
   if(EATradeAllowed)
      Print("Le commerce par les conseillers pour ce compte est permis");
   else
      Print("Le commerce par les conseillers pour ce compte est interdit!");

//--- découvrons le type de compte
   switch(tradeMode)
     {
      case(ACCOUNT_TRADE_MODE_DEMO):
         Print("C'est un compte de démonstration");
         break;
```

© 2000-2024, MetaQuotes Ltd.

```
      case(ACCOUNT_TRADE_MODE_CONTEST):
         Print("C'est un compte de compétition");
         break;
      default:Print("C'est un compte réel!");
     }

//--- déterminons la mode de la spécification du niveau StopOut
   switch(stopOutMode)
     {
      case(ACCOUNT_STOPOUT_MODE_PERCENT):
         Print("Le niveau StopOut est spécifiée en pourcentage");
         break;
      default:Print("Le niveau StopOut est spécifiée en moyens monétaires");
     }
  }
```

Voir aussi

Information sur le compte

© 2000-2024, MetaQuotes Ltd.

AccountInfoString

Rend la valeur de la propriété correspondante du compte.

```
string  AccountInfoString(
   ENUM_ACCOUNT_INFO_STRING  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. La valeur peut être une des valeurs ENUM_ACCOUNT_INFO_STRING.

La valeur rendue

La valeur du type string.

Exemple:

```
void OnStart()
  {
//--- montrons toute l'information accessible de la fonction AccountInfoString()
   Print("Nom du courtier = ",AccountInfoString(ACCOUNT_COMPANY));
   Print("Devise du dépôt = ",AccountInfoString(ACCOUNT_CURRENCY));
   Print("Nom du client = ",AccountInfoString(ACCOUNT_NAME));
   Print("Nom de serveur commercial = ",AccountInfoString(ACCOUNT_SERVER));
  }
```

Voir aussi

Information sur le compte

© 2000-2024, MetaQuotes Ltd.

La vérification de l'état

Les fonctions qui rendent des paramètres de l'état courant du terminal de client

Fonction	Action
GetLastError	Rend la valeur de la dernière erreur
IsStopped	Rend true, si on a donné l'ordre de terminer l'exécution du programme mql5
UninitializeReason	Rend le code de la raison de la déinitialisation
TerminalInfoInteger	Rend la valeur du type entier de la propriété correspondante de l'environnement du programme mql5
TerminalInfoDouble	Rend la valeur du type double de la propriété correspondante de l'environnement du programme mql5
TerminalInfoString	Rend la valeur du type string de la propriété correspondante de l'environnement du programme mql5
MQLInfoInteger	Rend la valeur du type entier de la propriété correspondante du programme mql5 lancé
MQLInfoString	Rend la valeur du type string de la propriété correspondante du programme mql5 lancé
Symbol	Rend le nom du symbole du graphique courant
Period	Rend la valeur du temps trame du graphique courant
Digits	Rend le nombre de signes décimaux après la virgule, définissant l'exactitude de la mesure du prix du symbole du graphique courant
Point	Rend la dimension de point de l'instrument courant en devise de la cotation

© 2000-2024, MetaQuotes Ltd.

GetLastError

Rend le contenu de la variable du système _LastError.

```
int  GetLastError();
```

La valeur rendue

Rend la valeur de la dernière erreur, qui s'est produite pendant l'exécution du programme mql5.

Note

Après l'appel de la fonction, le contenu de la variable _LastError ne se remet pas au zéro. Pour remettre au zéro cette variable il faut appeler cette fonction ResetLastError()

Voir aussi

Codes du retour du serveur commercial

© 2000-2024, MetaQuotes Ltd.

IsStopped

Vérifie la fermeture forcée du travail du programme mql5.

```
bool  IsStopped();
```

La valeur rendue

Rend true, si la variable du système _StopFlag contient la valeur, qui se diffère du 0. La valeur non zéro s'inscrit à la variable _StopFlag, si on a reçu l'ordre de terminer l'exécution du programme mql5. Dans ce cas il est nécessaire de terminer le travail du programme le plus vite possible, au contraire le programme sera terminé forcément du dehors dans 3 secondes.

© 2000-2024, MetaQuotes Ltd.

UninitializeReason

Rend le code de la raison de la déinitialisation.

```
int  UninitializeReason();
```

La valeur rendue

Rend la valeur de la variable _UninitReason, qui est formé avant l'appel de la fonction OnDeinit(). La valeur dépend de la raison qui a amené à la déinitialisation.

© 2000-2024, MetaQuotes Ltd.

TerminalInfoInteger

Rend la valeur d'une propriété correspondante de l'environnement de programme mql5.

```
int  TerminalInfoInteger(
   int  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. Peut être une des valeurs de l'énumération ENUM_TERMINAL_INFO_INTEGER.

La valeur rendue

La valeur du type int.

© 2000-2024, MetaQuotes Ltd.

TerminalInfoDouble

Возвращает значение соответствующего свойства окружения mql5-программы.

```
double  TerminalInfoDouble(
   int  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Может быть одним из значений перечисления ENUM_TERMINAL_INFO_DOUBLE.

Возвращаемое значение

Значение типа double.

© 2000-2024, MetaQuotes Ltd.

TerminalInfoString

La fonction rend la valeur de la propriété correspondante de l'environnement de programme mql5. La propriété doit être du type string

```
string  TerminalInfoString(
   int  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. Peut être une des valeurs de l'énumération ENUM_TERMINAL_INFO_STRING.

La valeur rendue

La valeur du type string.

© 2000-2024, MetaQuotes Ltd.

MQLInfoInteger

Rend la valeur de la propriété correspondante du programme MQL5 lancé.

```
int  MQLInfoInteger(
   int  property_id      //  identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. Peut être une des valeurs de l'énumération ENUM_MQL_INFO_INTEGER.

La valeur rendue

La valeur du type int.

© 2000-2024, MetaQuotes Ltd.

MQLInfoString

Rend la valeur de la propriété correspondante du programme MQL5 lancé.

```
string  MQLInfoString(
   int  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. Peut être une des valeurs de l'énumération ENUM_MQL_INFO_STRING.

La valeur rendue

La valeur du type string.

© 2000-2024, MetaQuotes Ltd.

Symbol

Rend le nom du symbole courant.

```
string  Symbol();
```

La valeur rendue

La valeur de la variable du système _Symbol, dans laquelle se trouve le nom du sumbole du graphique courant.

Note

Contrairement aux Expert Advisors, indicateurs et scripts, les services ne sont pas liés à un graphique spécifique. La fonction Symbol() retourne donc une chaîne vide ("") pour un service.

© 2000-2024, MetaQuotes Ltd.

Period

Rend la valeur du temps trame du graphique courant.

```
ENUM_TIMEFRAMES  Period();
```

La valeur rendue

Le contenu de la variable _Period, où se trouve la valeur du temps trame du graphique courant. La valeur peut être une des valeurs de l'énumération ENUM_TIMEFRAMES.

Note

Contrairement aux Expert Advisors, indicateurs et scripts, les services ne sont pas liés à un graphique spécifique. La fonction Period() retourne donc 0 pour un service.

Voir aussi

PeriodSeconds, Les périodes des graphiques, Date et temps, La visibilité des objets

© 2000-2024, MetaQuotes Ltd.

Digits

Rend le nombre de signes décimaux après la virgule, définissant l'exactitude de la mesure du prix du symbole du graphique courant.

```
int  Digits();
```

La valeur rendue

La valeur de la variable _Digits, qui conserve le nombre de chiffres décimaux après la virgule, déterminant l'exactitude de la mesure du prix du du symbole de graphique actuel.

© 2000-2024, MetaQuotes Ltd.

Point

Rend la dimension de point de l"instrument courant en devise de la cotation.

```
double  Point();
```

La valeur rendue

La valeur de la variable _Point, où se trouve la grandeur du point de l"instrument courant en devise de la cotation.

© 2000-2024, MetaQuotes Ltd.

Gestion d'Evènements

Le langage MQL5 fournit le traitement de certains évènements prédéfinis. Les fonctions pour gérer ces évènements doivent être définies dans le programme MQL5 : le nom de la fonction, le type de retour, l'ensemble de paramètres (s'il y en a) et leurs types doivent correspondre totalement à la description d'une fonction de gestion de l'évènement.

Le gestionnaire d'évènements du terminal client utilise le type de retour et les types des paramètres pour identifier les fonctions gérant un évènement. Si une certaine fonction a des paramètres ou un type de retour ne correspondant pas aux descriptions ci-dessous, cette fonction ne pourra pas être utilisée pour gérer un évènement.

Fonction	Description
OnStart	La fonction est appelée lorsque l'évènement Start se produit pour effectuer les actions définies dans le script
OnInit	La fonction est appelée dans les indicateurs et les EA lorsque l'évènement Init se produit pour initialiser un programme MQL5 lancé
OnDeinit	La fonction est appelée dans les indicateurs et les EA lorsque l'évèenement Deinit se produit pour désinitialiser un programme MQL5 lancé
OnTick	La fonction est appelée dans les EA lorsque l'évènement NewTick se produit pour gérer une nouvelle cotation
OnCalculate	La fonction est appelée dans les indicateurs lorsque l'évènement Calculate se produit pour gérer les changements de prix
OnTimer	La fonction est appelée dans les indicateurs et les EA pendant l'évènement périodique Timer généré par le terminal à des intervalles fixes de temps
OnTrade	La fonction est appelée dans les EA pendant l'évènement Trade généré à la fin d'une opération de trading sur un serveur de trades
OnTradeTransaction	La fonction est appelée dans les EA lorsque l'évènement TradeTransaction se produit pour traiter les résultats de l'exécution d'une demande de trade
OnBookEvent	La fonction est appelée dans les EA lorsque l'évènement BookEvent se produit pour traiter les changements du market depth
OnChartEvent	La fonction est appelée dans les indicateurs et les EA lorsque l'évènement ChartEvent se produit pour traiter les changements du graphique effectués par un utilisateur ou un programme MQL5
OnTester	La fonction est appelée dans les EA lorsque l'évènement Tester se produit pour effectuer les actions nécessaires après le test d'un EA sur les données historiques
OnTesterInit	La fonction est appelée dans les EA lorsque l'évènement TesterInit se produit pour effectuer les actions nécessaires avant une

© 2000-2024, MetaQuotes Ltd.

Fonction	Description
	optimisation dans le testeur de stratégies
OnTesterDeinit	La fonction est appelée dans les EA lorsque l'évènement TesterDeinit se produit après l'optimisation d'un EA dans le testeur de stratégie
OnTesterPass	La fonction est appelée dans les EA lorsque l'évènement TesterPass se produit pour gérer l'arrivée d'un nouvel ensemble de données pendant l'optimisation d'un EA dans le testeur de stratégie

Le terminal client envoie les évènements entrants vers les graphiques ouverts correspondants. Les évènements peuvent également être générés par les graphiques (évènements graphiques) ou par des programmes mql5 (évènements personnalisés). La génération des évènements de création/suppression des objets graphiques peut être activée/désactivée en définissant les propriétés du graphique CHART_EVENT_OBJECT_CREATE et CHART_EVENT_OBJECT_DELETE. Chaque application mql5 et chaque graphique ont leur propre queue d'évènements où chaque nouvel évènement est placé

Un programme ne reçoit les évènements que du graphique sur lequel il est en cours d'exécution. Tous les évènements sont gérés les uns après les autres dans l'ordre de réception. Si la queue contient déjà l'évènement NewTick ou que cet évènement est déjà en cours de traitement, alors le nouvel évènement NewTick n'est pas ajouté dans la queue de l'application mql5. De la même façon, si le ChartEvent est déjà dans la queue d'un programme mql5 ou qu'un tel évènement est en cours de traitement, alors tout nouvel évènement de ce type ne sera pas placé dans la queue. La gestion des évènements du time est effectuée de la même façon – si l'évènement Timer est déjà dans la queue ou est en cours de traitement, aucun nouvel évènement timer n'est ajouté dans la queue.

Les queues d'évènements ont une taille limitée mais suffisante, donc un overflow de la queue est peu probable dans un programme développé correctement. Lorsqu'un overflow de la queue se produit, les nouveaux évènements sont annulés sans avoir été ajouté dans la queue.

Il est fortement recommandé de ne pas utiliser de boucle infinie pour gérer les évènements. Les exceptions possibles sont les scripts gérant un seul évènement Start.

Les bibliothèques ne gérent aucun évènement.

© 2000-2024, MetaQuotes Ltd.

OnStart

La fonction est appelée dans les EA lorsque l'évènement Start apparaît. La fonction est prévue pour l'exécution unique d'actions implémentées dans le script. Il y a 2 types de fonctions.

Version retournant un résultat

```
int  OnStart(void);
```

Valeur de Retour

La valeur de type int affichée dans l'onglet Journal.

L'entrée "script nom_du_script removed (result code N)" est créée dans le journal du terminal après la fin de l'exécution du script. Ici N est une valeur retournée par la fonction OnStart().

L'entrée "service nom_du_service stopped (result code N)" est créée dans le journal du terminal après la fin de l'exécution du service. Ici N est une valeur retournée par la fonction OnStart().

Il est recommandé d'utiliser l'appel OnStart() retournant le résultat de l'exécution puisqu'il permet non seulement l'exécution d'un script ou d'un service, mais retourne également un code d'erreur et d'autres données utiles pour analyser le résultat de l'exécution du programme.

La version sans retour de résultat n'est utilisé que pour la compatibilité avec les anciens codes. Il n'est pas recommandé de l'utiliser

```
void  OnStart(void);
```

Note

OnStart() est la seule fonction pour gérer des évènements dans les scripts et dans les services. Aucun autre évènement n'est envoyé à ces programmes. A son tour, l'évènement Start n'est pas passé aux EA et aux indicateurs personnalisés.

SCript exemple :

```
//--- macros utilisant les couleurs
#define XRGB(r,g,b)     (0xFF000000|(uchar(r)<<16)|(uchar(g)<<8)|uchar(b))
#define GETRGB(clr)     ((clr)&0xFFFFFF)
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- définit la couleur des bougies baissières
   Comment("Définit la couleur des bougies baissières");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BEAR,GetRandomColor());
   ChartRedraw(); // met à jour le graphique immédiatement sans attendre un nouveau t
   Sleep(1000);   // effectue une pause d'1 seconde pour voir les changements
//--- définit la couleur des bougies haussières
   Comment("Définit la couleur des bougies haussières");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BULL,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit la couleur d'arrière plan
   Comment("Définit la couleur d'arrière plan");
   ChartSetInteger(0,CHART_COLOR_BACKGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- définit la couleur de la ligne Ask
   Comment("Définit la couleur de la ligne Ask");
   ChartSetInteger(0,CHART_COLOR_ASK,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- définit la couleur de la ligne Bid
   Comment("Définit la couleur de la ligne Bid");
   ChartSetInteger(0,CHART_COLOR_BID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- définit la couleur d'une barre baissière et du contour d'une bougie baissière
   Comment("Définit la couleur d'une barre baissière et du contour d'une bougie baissi
   ChartSetInteger(0,CHART_COLOR_CHART_DOWN,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- définit la couleur de la ligne du graphique en courbre et des bougies Dojo
   Comment("Définit la couleur de la ligne du graphique en courbre et des bougies Dojo
   ChartSetInteger(0,CHART_COLOR_CHART_LINE,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- définit la couleur d'une barre haussière et du contour d'une bougie haussière
   Comment("Définit la couleur d'une barre haussière et du contour d'une bougie haussi
   ChartSetInteger(0,CHART_COLOR_CHART_UP,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- définit la couleur des axes, des échelles et de la ligne OHLC
   Comment("Définit la couleur des axes, des échelles et de la ligne OHLC");
   ChartSetInteger(0,CHART_COLOR_FOREGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- définit la couleur de la grille
   Comment("Définit la couleur de la grille");
   ChartSetInteger(0,CHART_COLOR_GRID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- définit la couleur du prix Last
   Comment("Définit la couleur du prix Last");
   ChartSetInteger(0,CHART_COLOR_LAST,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- définit la couleur des niveaux des ordres Stop Loss et Take Profit
   Comment("Définit la couleur des niveaux des ordres Stop Loss et Take Profit");
   ChartSetInteger(0,CHART_COLOR_STOP_LEVEL,GetRandomColor());
   ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
   Sleep(1000);
//--- définit la couleur des volumes et des niveaux d'entrée sur le marché
   Comment("Définit la couleur des volumes et des niveaux d'entrée sur le marché");
   ChartSetInteger(0,CHART_COLOR_VOLUME,GetRandomColor());
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Retourne une couleur générée aléatoirement                       |
//+------------------------------------------------------------------+
color GetRandomColor()
  {
   color clr=(color)GETRGB(XRGB(rand()%255,rand()%255,rand()%255));
   return clr;
  }
```

Voir également

Fonctions de gestion des évènements, Exécution du programme, Evènements du terminal client

© 2000-2024, MetaQuotes Ltd.

OnInit

La fonction est appelée dans les indicateurs et les EA lorsque l'évènement Init se produit. Il est utilisé pour initialiser un programme MQL5 en cours d'exécution. Il y a 2 types de fonctions.

La **version qui retourne le résultat**

```
int  OnInit(void);
```

Valeur de Retour

Une valeur de type int, zéro signifie une initialisation réussie.

L'appel OnInit() qui retourne le résultat de l'exécution est recommandé puisqu'il permet non seulement l'initialisation du programme, mais retourne également un code d'erreur dans le cas d'une fin prématurée du programme.

La version sans retour de résultat n'est utilisé que pour la compatibilité avec les anciens codes. Il n'est pas recommandé de l'utiliser

```
void  OnInit(void);
```

Note

L'évèenement Init est généré immédiatement après le chargement d'un EA ou d'un indicateur. L'évènement n'est pas généré pour les scripts. La fonction OnInit() est utilisée pour initialiser un programme MQL5. Si OnInit() a une valeur de retour de type int, un code de retour différent de zéro signifie un échec de l'initialisation et génère l'évènement Deinit avec le code de raison de désinitialisation REASON_INITFAILED.

La fonction OnInit() de type void signifie toujours une initialisation réussie et il n'est pas recommandé de l'utiliser.

Pour optimiser les entrées d'un EA, il est recommandé d'utiliser les valeurs de l'énumération ENUM_INIT_RETCODE comme code de retour. Ces variables sont prévues pour gérer le procédé d'optimisation, incluant la sélection des agents de test les plus pertinents. Il est possible de demander les données de la configuration d'un agent et des ressources (nombre de coeurs, quantité de mémoire libre, etc.) en utilisant la fonction TerminalInfoInteger() pendant l'initialisation de l'EA avant le lancement du test. Sur la base des données obtenues, vous pouvez soit permettre d'utiliser l'agent de test ou le bannir de l'optimisation de l'EA.

ID	Description
INIT_SUCCEEDED	Initialisation réussie, le test de l'EA peut continuer. Ce code signifie la même chose que la valeur zéro - l'initialisation de l'EA dans le testeur est réussie.
INIT_FAILED	Initialisation échouée. Il n'y a pas de raison de continuer le test en raison d'erreurs inévitables. Par exemple, il est impossible de créer un indicateur nécessaire pour l'opération de l'EA. Le retour de cette valeur signifie la même chose que le retour d'une valeur différente de zéro - l'initialisation de l'EA dans le testeur a échouée.

© 2000-2024, MetaQuotes Ltd.

ID	Description
INIT_PARAMETERS_INCORRECT	Conçu pour dénoter un ensemble incorrect des paramètres d'entrée par un développeur. Dans le tableau d'optimisation générale, la chaîne de caractères résultat avec ce code de retour est mise en surbrillance en rouge. Un test pour un tel ensemble d'entrées de l'EA n'est pas effectué. L'agent est prêt à recevoir une nouvelle tâche. Lorsque cette valeur est reçue, le testeur de stratégie ne passe pas cette tâche aux autres agents pour une exécution répétée.
INIT_AGENT_NOT_SUITABLE	Aucune erreur d'exécution du programme pendant l'initialisation. Cependant, pour certaines raisons, l'agent ne convient pas pour effectuer un test. Par exemple, il n'y a pas assez de RAM, pas de support d'OpenCL, etc. Après avoir retourné ce code, l'agent ne reçoit plus de tâches jusqu'à la fin de cette optimisation.

Utiliser OnInit() retournant INIT_FAILED/INIT_PARAMETERS_INCORRECT dans le tester a certaines spécificités qu'il faut prendre en compte lors de l'optimisation des EA :

- l'ensemble de paramètres pour lequel OnInit() a retourné INIT_PARAMETERS_INCORRECT est considéré invalide pour le test et n'est pas utilisé pour obtenir la population suivante pendant une optimisation génétique. De trop nombreux ensembles de paramètres "écartés" conduisent à des résultats incorrects lors de la recherche des paramètres optimaux de l'EA. L'algorithme de recherche assume que la fonction du critère d'optimisation est lissée et n'a pas de trou sur l'intervalle entier des paramètres d'entrée.
- si OnInit() retourne INIT_FAILED, cela signifie qu'un test ne peut pas être lancé, et l'EA est enlevé de la mémoire de l'agent. L'EA est rechargé à nouveau pour effectuer la prochaine passe avec un nouvel ensemble de paramètres. Le lancement de la passe d'optimisation suivante prend plus de temps comparé à appeler TesterStop().

Exemple de fonction OnInit() pour un EA

```
//--- input parameters
input int      ma_period=20; // période de la moyenne mobile

//--- handle de l'indicateur utilisé dans l'EA
int indicator_handle;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vérifie la validité de ma_period
   if(ma_period<=0)
     {
      PrintFormat("Valeur d'entrée ma_period invalide : %d",ma_period);
      return (INIT_PARAMETERS_INCORRECT);
     }
//--- pendant l'optimisation
   if(MQLInfoInteger(MQL_OPTIMIZATION))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- vérifie la RAM disponible pour l'agent
       int available_memory_mb=TerminalInfoInteger(TERMINAL_MEMORY_TOTAL);
       if(available_memory_mb<2000)
         {
          PrintFormat("Mémoire insuffisante pour l'agent de test : %d MB",
                      available_memory_mb);
          return (INIT_AGENT_NOT_SUITABLE);
         }
      }
//--- vérifie l'indicateur
   indicator_handle=iCustom(_Symbol,_Period,"My_Indicator",ma_period);
   if(indicator_handle==INVALID_HANDLE)
      {
       PrintFormat("Echec de la génération du handle sur My_Indicator. Code d'erreur %d
                   GetLastError());
       return (INIT_FAILED);
      }
//--- Initialisation de l'EA réussie
   return(INIT_SUCCEEDED);
  }
```

Voir également

OnDeinit, Fonctions de gestion des évènements, Exécution du programme, Evènements du terminal client, Initialisation des variables, Créer et supprimer des objets

© 2000-2024, MetaQuotes Ltd.

OnDeinit

La fonction est appelée dans les indicateurs et les EA lorsque l'évènement Deinit se produit. Il est utilisé pour désinitialiser un programme MQL5 en cours d'exécution.

```
void  OnDeinit(
   const int  reason         // code de la raison de la désinitialisation
   );
```

Parameters

reason

[in] Code de la raison de la désinitialisation.

Valeur de Retour

Aucune valeur de retour

Note

L'évènement Deinit est généré pour les EA et les indicateurs dans les cas suivants :

- avant une réinitialisation en raison du changement du symbole ou de la période du graphique auquel le programme mql5 est attaché ;
- avant une réinitialisation en raison du changement des entrées ;
- avant le déchargement d'un programme mql5.

Le paramètre *reason* peut avoir les valeurs suivantes :

Constante	Valeur	Description
REASON_PROGRAM	0	L'EA a cessé de fonctionner en appelant la fonction ExpertRemove()
REASON_REMOVE	1	Programme enlevé du graphique
REASON_RECOMPILE	2	Programme recompilé
REASON_CHARTCHANGE	3	Symbole ou période du graphique modifié
REASON_CHARTCLOSE	4	Graphique fermé
REASON_PARAMETERS	5	Entrées modifiées par l'utilisateur
REASON_ACCOUNT	6	Un autre compte a été activé ou une reconnexion au serveur de trades s'est produite en raison de changements dans les paramètres du compte
REASON_TEMPLATE	7	Un autre modèle de graphique a été appliqué
REASON_INITFAILED	8	La fonction OnInit() a retourné une valeur différente de 0
REASON_CLOSE	9	Terminal fermé

Les codes des raisons de désinitialisation d'un EA peuvent être récupérés avec la fonction UninitializeReason() ou avec la variable prédéfinie _UninitReason.

© 2000-2024, MetaQuotes Ltd.

Exemple des fonctions OnInit() et OnDeinit() pour l'EA

```
input int fake_parameter=3;      // paramètre inutile
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Récupère le numéro du buid où le programme est compilé
   Print(__FUNCTION__," Build #",__MQLBUILD__);
//--- Le code de la raison de la réinitialisation peut être obtenu dans OnInit()
   Print(__FUNCTION__," Le code de la raison de la désinitialisation peut être obtenu
//--- Première façon de récupérer le code de la raison de la désinitialisation
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Deuxième façon de récupérer le code de la raison de la désinitialisation
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de dé-initialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Première façon de récupérer le code de la raison de la désinitialisation
   Print(__FUNCTION__," Code de la raison de la désinitialisation = ",reason);
//--- Deuxième façon de récupérer le code de la raison de la désinitialisation
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Troisième façon de récupérer le code de la raison de la désinitialisation
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
  }
//+------------------------------------------------------------------+
//| Retourne une description du code de la désinitialisation         |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Le compte a été changé";break;
      case REASON_CHARTCHANGE:
         text="Le symbole ou la période ont été changés";break;
      case REASON_CHARTCLOSE:
         text="Le graphique a été fermé";break;
      case REASON_PARAMETERS:
         text="Les paramètres d'entrée ont été changés";break;
      case REASON_RECOMPILE:
```

© 2000-2024, MetaQuotes Ltd.

```
         text="Le programme "+__FILE__+" a été recompilé";break;
      case REASON_REMOVE:
         text="Le programme "+__FILE__+" a été enlevé du graphique";break;
      case REASON_TEMPLATE:
         text="Un nouveau modèle a été appliqué au graphique";break;
      default:text="Autre raison";
     }
//---
   return text;
  }
```

Voir également

OnInit, Fonctions de gestion des évènements, Exécution du programme, Evènements du terminal client, Codes des raisons de la non initialisation, Portée de visibilité et cycle de vie des variables, Créer et supprimer des objets

© 2000-2024, MetaQuotes Ltd.

OnTick

La fonction est appelée dans les EA lorsque l'évènement NewTick se produit pour gérer une nouvelle cotation.

```
void  OnTick(void);
```

Valeur de Retour

Aucune valeur de retour

Note

L'évènement NewTick n'est généré que pour les EA lorsqu'un nouveau tick est reçu pour le symbole auquel l'Expert Advisor est attaché. Il est inutile de définir la fonction OnTick() dans un indicateur personnalisé ou dans un script puisque l'évènement NewTick n'est pas généré pour eux.

L'évènement Tick n'est généré que pour les Expert Advisors, mais cela ne signifie pas que les Expert Advisors ont besoin de la fonction OnTick(), puisque les évènements Timer, BookEvent et ChartEvent sont également générés pour les Expert Advisors en plus de NewTick.

Tous les évènements sont gérés les uns après les autres dans l'ordre de réception. Si la queue contient déjà l'évènement NewTick ou que cet évènement est déjà en cours de traitement, alors le nouvel évènement NewTick n'est pas ajouté dans la queue de l'application mql5.

L'évènement NewTick est généré indépendamment du mode d'auto-trading. Le trading automatique désactivé signifie l'interdiction d'envoyer des demandes de trade depuis un EA. Le fonctionnement de l'EA n'est pas stoppé.

Désactier le trading automatique en appuyant sur le bouton AutoTrading n'interrompt pas l'exécution de la fonction OnTick() en cours.

Exemple d'un EA avec sa logique de trading entièrement dans la fonction OnTick()

```
//+------------------------------------------------------------------+
//|                                                   TradeByATR.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "EA exemple tradant dans la direction de la bougie \"explosive\"
#property description "Une bougie "Explosive\" a la taille de son corps supérieure à
#property description "Le paramètre \"revers\" renverse la direction du signal"

input double lots=0.1;         // volume en lots
input double kATR=3;           // longueur de la bougie de signal dans l'ATR
input int    ATRperiod=20;     // période de l'ATR
input int    holdbars=8;       // nombre de barre pour tenir la position
input int    slippage=10;      // slippage autorisé
input bool   revers=false;     // renversement de signal ?
input ulong  EXPERT_MAGIC=0;   // nombre magique de l'EA
```

© 2000-2024, MetaQuotes Ltd.

```
//--- pour stocker le handle de l'ATR
int atr_handle;
//--- nous stockerons ici les dernières valeurs de l'ATR et le corps de la bougie
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- initialisation des variables globales
   last_atr=0;
   last_body=0;
//--- définit le volume correct
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- crée le handle de l'indicateur ATR
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: échec de la création de iATR, code d'erreur %d",__FUNCTION__,Ge
      return(INIT_FAILED);
     }
//--- initialisation réussie de l'EA
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de dé-initialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- informe du code de fin de l'exécution de l'EA
   Print(__FILE__,": Raison de la désinitialisation = ",reason);
  }
//+------------------------------------------------------------------+
//| Fonction de tick de l'Expert                                     |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- signal de trading
   static int signal=0; // +1 signifie un signal d'achat, -1 signifie un signal de ver
//--- vérifie et ferme les anciennes positions ouvertes plus de 'holdbars' barres aupa
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- vérifie si c'est une nouvelle barre
   if(isNewBar())
     {
      //--- vérifie la présence d'un signal
      signal=CheckSignal();
```

```
      }
//--- si une position de type netting est ouverte, ignore le signal - attends jusqu'à
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
      {
       signal=0;
       return; // sors de la fonction NewTick et n'entre pas sur le marché avant qu'une
      }
//--- pour un compte de type hedging, chaque position est conservée et fermée séparéme
   if(signal!=0)
      {
       //--- signal d'achat
       if(signal>0)
         {
          PrintFormat("%s: Signal d'achat ! Revers=%s",__FUNCTION__,string(revers));
          if(Buy(trade_lot,slippage,EXPERT_MAGIC))
             signal=0;
         }
       //--- signal de vente
       if(signal<0)
         {
          PrintFormat("%s: Signal de vente ! Revers=%s",__FUNCTION__,string(revers));
          if(Sell(trade_lot,slippage,EXPERT_MAGIC))
             signal=0;
         }
      }
//--- fin de la fonction OnTick
  }
//+------------------------------------------------------------------+
//| Vérifie la présence d'un nouveau signal de trading               |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0 signifie aucun signal
   int res=0;
//--- récupère la valeur de l'ATR sur l'avant dernière barre (l'indice de la barre est
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
      {
       last_atr=atr_value[0];
       //--- récupère les données de la dernière barre fermée dans le tableau d'élément
       MqlRates bar[1];
       if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
         {
          //--- calcule la taille du corps de la dernière barre complétée
          last_body=bar[0].close-bar[0].open;
          //--- si le corps de la dernière barre (avec l'indice 1) est supérieur à la v
          if(MathAbs(last_body)>kATR*last_atr)
             res=last_body>0?1:-1; // valeur positive pour une bougie haussière
         }
```

```
      else
         PrintFormat("%s: Impossible de recevoir la dernière barre ! Erreur",__FUNCTIO
      }
   else
      PrintFormat("%s: Impossible de recevoir la valeur de l'ATR ! Erreur",__FUNCTION_
//--- si le mode de trading renversé est activé
   res=revers?-res:res;  // retourne le signal si nécessaire (retourne -1 au lieu de 1
//--- retourne la valeur du signal de trading
   return (res);
  }
//+------------------------------------------------------------------+
//|  Retourne 'true' lorsqu'une nouvelle barre apparaît              |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // stocke l'heure d'ouverture de la barre courante
//--- récupère l'heure d'ouverture de la barre zéro
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- si l'heure d'ouverture change, une nouvelle barre est arrivée
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- affiche les données de l'heure d'ouverture de la nouvelle barre dans le jo
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- affiche un message avec l'heure d'ouverture de la nouvelle barre
         PrintFormat("%s: nouvelle barre sur %s %s ouverte à %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- récupère les données du dernier tick
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("Echec de SymbolInfoTick(), erreur = ",GetLastError());
         //--- affiche l'heure du dernier tick en millisecondes
         PrintFormat("Le dernier tick était à %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- nous avons une nouvelle barre
      return (true);
     }
//--- aucune nouvelle barre
   return (false);
  }
//+------------------------------------------------------------------+
//| Achète au prix du marché avec le volume spécifié                 |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
```

```
//--- achète au prix du marché
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Vends au prix du marché avec le volume spécifié                  |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- vends au prix du marché
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Ferme les positions détenues trop longtemps (en barres)          |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // nombre de positions ouvertes
//--- itère sur les positions ouvertes
   for(int i=total-1; i>=0; i--)
     {
      //--- paramètres de la position
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      datetime position_open=(datetime)PositionGetInteger(POSITION_TIME);
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;

      //--- si la position est trop vieille et que le MagicNumber et le symbole corres
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
         double volume=PositionGetDouble(POSITION_VOLUME);
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // mets le texte en minuscules
         PrintFormat("Ferme la position #%d %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- définit le type de l'ordre et envoie une demande de trade
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Prépare et envoie une demande de trade                           |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- déclare et initialise les structures
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- paramètres de la demande
   request.action   =TRADE_ACTION_DEAL;                     // trading operation type
   request.position =pos_ticket;                            // position ticket if clos
   request.symbol   =Symbol();                              // symbole
   request.volume   =volume;                                // volume
   request.type     =type;                                  // type de l'ordre
   request.price    =price;                                 // prix du trade
   request.deviation=slip;                                  // déviation du prix autor
   request.magic    =magicnumber;                           // MagicNumber de l'ordre
//--- envoie une demande
   if(!OrderSend(request,result))
     {
      //--- affiche les informations de l'échec
      PrintFormat("OrderSend %s %s %.2f à %.5f, erreur %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- information sur l'opération réussie
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Voir également

Fonctions de gestion des évènements, Exécution du programme, Evènements du terminal client, OnTimer, OnBookEvent, OnChartEvent

© 2000-2024, MetaQuotes Ltd.

OnCalculate

La fonction est appelée dans les indicateurs lorsque l'évènement Calculate se produit pour gérer les changements de prix. Il y a 2 types de fonctions. Un seul peut être utilisé dans un indicateur simple.

Calculs basés sur un tableau de données

```
int  OnCalculate(
   const int        rates_total,       // taille du tableau price[]
   const int        prev_calculated,   // nombre de barres gérées durant l'appel précé
   const int        begin,             // numéro de l'indice dans le tableau price[] c
   const double&    price[]            // tableau de valeurs pour le calcul
   );
```

Calculs basés sur les séries de la période courante

```
int  OnCalculate(
   const int        rates_total,       // taille des séries temporelles d'entrée
   const int        prev_calculated,   // nombre de barres gérées durant l'appel précé
   const datetime&  time[],            // Tableau des heures
   const double&    open[],            // Tableau des prix Open
   const double&    high[],            // Tableau des prix High
   const double&    low[],             // Tableau des prix Low
   const double&    close[],           // Tableau des prix Close
   const long&      tick_volume[],     // Tableau du volume des ticks
   const long&      volume[],          // Tableau des volumes réels
   const int&       spread[]           // Tableau des spreads
   );
```

Parameters

rates_total

[in] Taille du tableau price[] ou des séries d'entrée disponibles pour l'indicateur pour le calcul. Dans le 2ème type de fonction, la valeur du paramètre correspond au nombre de barres du graphique sur lequel il est lancé.

prev_calculated

[in] Contient la valeur retournée par la fonction OnCalculate() pendant l'appel précédent. Il est conçu pour ignorer les barres qui n'ont pas changé depuis le lancement précédent de cette fonction.

begin

[in] Indice des données du tableau price[] à partir duquel les données utiles commencent. Il vous permet d'ignorer les données initiales ou les données manquantes, pour lesquelles il n'y a pas de valeurs correctes.

price[]

[in] Tableau de valeurs pour les calculs. L'une des timeseries des prix ou le buffer d'un indicateur calculé peuvent être passés comme tableau price[]. Le type des données passées pour le calcul peut être défini avec la variable prédéfinie _AppliedTo.

time[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau contenant les heures d'ouverture des barres.

open[]

[in] Tableau contenant les prix Open des barres.

high[]

[in] Tableau contenant les prix High des barres.

low[]

[in] Tableau contenant les prix Low des barres.

close[]

[in] Tableau contenant les prix Close des barres.

tick_volume[]

[in] Tableau contenant les volumes des ticks.

volume[]

[in] Tableau contenant les volumes réels.

spread[]

[in] Tableau contenant les spreads des barres.

Valeur de Retour

Valeur de type int à passer dans le paramètre *prev_calculated* au prochain appel de la fonction.

Note

Si la fonction OnCalculate() est égale à zéro, aucune valeur pour l'indicateur n'est affichée dans la fenêtre des Données du terminal client.

Si le prix a changé depuis le dernier appel à la fonction OnCalculate() (un historique plus profond a été chargé ou des sauts dans l'hisrtorique ont été remplis), la valeur du paramètre d'entrée *prev_calculated* est mise à zéro par le terminal lui-même.

Pour définir la direction d'indexation dans les tableaux *time[]*, *open[]*, *high[]*, *low[]*, *close[]*, *tick_volume[]*, *volume[]* et *spread[]*, appelez la fonction ArrayGetAsSeries(). Pour ne pas dépendre des valeurs par défaut, appelez la fonction ArraySetAsSeries() pour les travaux sur lesquels vous souhaitez travailler.

Lors de l'utilisation du premier type de la fonction, une timeserie ou un indicateur est sélectionné par l'utilisateur comme tableau price[] dans l'onglet Paramètres lors du lancement de l'indicateur. Pour cela, spécifiez l'élément nécessaire dans la liste déroulante dans le champ "Appliquer à".

Pour obtenir les valeurs d'un indicateur personnalisé depuis d'autres programmes mql5, la fonction iCustom() est utilisée. Elle retourne un handle sur l'indicateur pour être utilisé par la suite. Il est également possible de spécifier le tableau *price []* ou le handle d'un autre indicateur. Ce paramètre doit être passé en dernier dans la liste des variables d'entrée d'un indicateur personnalisé.

Il est nécessaire d'utiliser le lien entre la valeur retournée par la fonction OnCalculate() et le deuxième paramètre d'entrée *prev_calculated*. Lors de l'appel à la fonction, le paramètre *prev_calculated* contient la valeur retournée par la fonction OnCalculate() pendant l'appel précédent. Ceci permet d'implémenter des algorithmes permettant d'économiser les ressources pour calculer un

© 2000-2024, MetaQuotes Ltd.

indicateur personnalisé afin d'éviter les calculs répétitifs pour les barres qui n'ont pas changé depuis l'appel précédent à cette fonction.

Indicateur exemple

```
//+------------------------------------------------------------------+
//|                                          OnCalculate_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Extrait du calcul de l'indicateur Momentum"

//---- paramètres de l'indicateur
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_type1   DRAW_LINE
#property indicator_color1  Blue
//---- entrées
input int MomentumPeriod=14; // Période du calcul
//---- buffer de l'indicateur
double    MomentumBuffer[];
//--- variable globale pour stocker la période de calcul
int       IntPeriod;
//+------------------------------------------------------------------+
//| Fonction d'initialisation d'un indicateur personnalisé           |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- vérification des paramètres d'entrée
   if(MomentumPeriod<0)
     {
      IntPeriod=14;
      Print("Le paramètre de la période a une valeur incorrecte. La valeur suivante es
     }
   else
      IntPeriod=MomentumPeriod;
//---- buffers
   SetIndexBuffer(0,MomentumBuffer,INDICATOR_DATA);
//---- nom de l'indicateur à afficher dans la fenêtre des Données et dans les sous-fen
   IndicatorSetString(INDICATOR_SHORTNAME,"Momentum"+"("+string(IntPeriod)+")");
//--- définit l'indice de la barre de début de dessin
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,IntPeriod-1);
//--- définit 0.0 comme valeur vide qui ne sera pas dessinée
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
```

```
//--- précision de l'affichage de l'indicateur
   IndicatorSetInteger(INDICATOR_DIGITS,2);
  }
//+------------------------------------------------------------------+
//|  Calcul de l'indicateur Momentum                                 |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,     // taille du tableau price[]
                const int prev_calculated, // nombre de barres gérées précédemment
                const int begin,           // indice du début des données significati
                const double &price[])     // tableau des valeurs
  {
//--- position initiale pour les calculs
   int StartCalcPosition=(IntPeriod-1)+begin;
//---- si les données de calcul sont insuffisantes
   if(rates_total<StartCalcPosition)
      return(0);  // sort avec la valeur zéro - l'indicateur n'est pas calculé
//--- début du dessin de l'indicateur
   if(begin>0)
      PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,StartCalcPosition+(IntPeriod-1));
//--- début des calculs, définit la position de départ
   int pos=prev_calculated-1;
   if(pos<StartCalcPosition)
      pos=begin+IntPeriod;
//--- boucle principale de calcul
   for(int i=pos;i<rates_total && !IsStopped();i++)
      MomentumBuffer[i]=price[i]*100/price[i-IntPeriod];
//--- exécution de OnCalculate terminée. Retourne la nouvelle valeur de prev_calculate
   return(rates_total);
  }
```

Voir également

ArrayGetAsSeries, ArraySetAsSeries, iCustom, Fonctions de gestion des évènements, Exécution du programme, Evènements du terminal client, Accès aux timeseries et aux indicateurs

© 2000-2024, MetaQuotes Ltd.

OnTimer

La fonction est appelée dans les indicateurs pendant l'évènement Timer généré par le terminal à des intervalles fixes de temps.

```
void  OnTimer(void);
```

Valeur de Retour

Aucune valeur de retour

Note

L'évènement Timer est générée périodiquement par le terminal client pour un EA, qui a activé le timer en utilisant la fonction EventSetTimer(). Habituellement, cette fonction est appelée dans la fonction OnInit(). Lorsque l'EA s'arrête de fonctionner, le timer doit être éliminé en utilisant EventKillTimer(), qui est habituellement appelé dans la fonction OnDeinit().

Chaque Expert Advisor et chaque indicateur fonctionne avec son propre timer recevant les évènements uniquement de ce timer. Pendant l'arrêt d'une application mql5, le timer est détruit obligatoirement dans le cas où il a été créé mais n'a pas été désactivé par la fonction EventKillTimer().

Si vous avez besoin de recevoir les évènements du timer plus d'une fois par seconde, utilisez EventSetMillisecondTimer() pour créer un timer haute résolution.

En général, lorsque la période du timer est réduite, le temps de test est augmenté puisque le handler des évènements du timer est appelé plus souvent. Lors du travail en temps réel, les évènements du timer ne sont pas générés plus d'une fois toutes les 10-16 millisecondes en raison des limitations matérielles.

Un seul timer peut être lancé pour chaque programme. Chaque application mql5 et chaque graphique ont leur propre queue d'évènements où tous les nouveaux évènements sont placés. Si la queue contient déjà l'évènement Timer ou que cet évènement est déjà en cours de traitement, alors le nouvel évènement NewTick n'est pas ajouté dans la queue de l'application mql5.

EA exemple avec la fonction OnTimer()

```
//+------------------------------------------------------------------+
//|                                                OnTimer_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Exemple d'utilisation du timer pour calculer l'heure du serveu
#property description "Il est recommandé d'exécuter l'EA à la fin d'une semaine de tra
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
```

```
//--- crée un timer avec une période de 1 seconde
   EventSetTimer(1);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de dé-initialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- détruit le timer après la fin du travail
   EventKillTimer();

  }
//+------------------------------------------------------------------+
//| Fonction de tick de l'Expert                                     |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Fonction du timer                                                |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- heure du premier appel à OnTimer()
   static datetime start_time=TimeCurrent();
//--- heure du serveur de trades au premier appel à OnTimer() call
   static datetime start_tradeserver_time=0;
//--- heure calculée du serveur de trades
   static datetime calculated_server_time=0;
//--- heure locale du PC
   datetime local_time=TimeLocal();
//--- heure estimée courante du serveur de trades
   datetime trade_server_time=TimeTradeServer();
//--- si l'heure du serveur est inconnue pour n'importe quelle raison, termine en avar
   if(trade_server_time==0)
      return;
//--- si la valeur initiale du serveur de trades n'est pas encore définie
   if(start_tradeserver_time==0)
     {
      start_tradeserver_time=trade_server_time;
      //--- définie une valeur calculée du serveur de trades
      Print(trade_server_time);
      calculated_server_time=trade_server_time;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   else
     {
      //--- augmente l'heure du premier appel à OnTimer()
      if(start_tradeserver_time!=0)
         calculated_server_time=calculated_server_time+1;;
     }
//---
   string com=StringFormat("                  Heure de début : %s\r\n",TimeToString(st
   com=com+StringFormat("                  Heure locale : %s\r\n",TimeToString(local_t
   com=com+StringFormat("Heure TimeTradeServer : %s\r\n",TimeToString(trade_server_tir
   com=com+StringFormat(" Heure estimée du serveur : %s\r\n",TimeToString(calculated_s
//--- affiche les valeurs de tous les compteurs du graphique
   Comment(com);
  }
```

Voir également

EventSetTimer, EventSetMillisecondTimer, EventKillTimer, GetTickCount, GetMicrosecondCount, Evènements du terminal client

© 2000-2024, MetaQuotes Ltd.

OnTrade

La fonction est appelée dans les EA lorsque l'évènement Trade apparaît. La fonction est prévue pour le traitement des changements dans les listes des ordres, des positions et des trades.

```
void  OnTrade(void);
```

Valeur de Retour

Aucune valeur de retour

Note

OnTrade() n'est appelée que pour les Expert Advisors. Elle n'est pas utilisée dans les indicateurs et dans les scripts même si vous ajoutez une fonction avec le même nom et le même type.

Pour n'importe quelle action de trading (placer un ordre en attente, ouvrir/fermer une position, placer des stops, activer des ordres en attente, etc.), l'historique des ordres et des trades et/ou la liste des positions et des ordres courants est changé de façon correspondante.

Lors du traitement d'un ordre, le serveur de trade envoie un message au terminal à propos de l'évènement Trade entrant. Pour récupérer les données correspondantes des ordres et des trades de l'historique, il est nécessaire d'effectuer une demande de l'historique de trading en utilisant d'abord HistorySelect().

Les évènements de trading sont générés par le serveur dans les cas suivants :

- changement des ordres actifs,
- changement des positions,
- changement des transactions,
- changement de l'historique de trading.

Chaque évènement Trade peut être le résultat d'une ou plusieurs demandes de trades. Les demandes de trades sont envoyées au serveur en utilisant OrderSend() ou OrderSendAsync(). Chaque demande peut aboutir à plusieurs évènements de trading. Vous ne pouvez pas vous fier à la phrase "Une demande - un évènement Trade", puisque le traitement des évènements peut être effectué en plusieurs étapes, et chaque opération peut changer l'état des ordres, des positions et de l'historique de trading.

La fonction OnTrade() est appelée après les appels correpondants OnTradeTransaction() calls. En général, il n'y a pas de corrélation exacte entre le nombre d'appels à OnTrade() et à OnTradeTransaction(). Un appel à OnTrade() correspond à un ou plusieurs appels à OnTradeTransaction.

EA exemple avec la fonction OnTrade()

```
//+------------------------------------------------------------------+
//|                                                OnTrade_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
input    int days=7;            // profondeur de l'historique de trading en jours
//--- définit les limites de l'historique de trading au niveau global
datetime     start;             // date de début de l'historique de trading dans le ca
datetime     end;               // date de fin de l'historique de trading dans le cach
//--- compteurs globaux
int          orders;            // nombre d'ordres actifs
int          positions;         // nombre de positions ouvertes
int          deals;             // nombre de transactions dans le cache de l'historiqu
int          history_orders;    // nombre d'ordres dans le cache de l'historique de tr
bool         started=false;     // flag de pertinence du compteur

//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   end=TimeCurrent();
   start=end-days*PeriodSeconds(PERIOD_D1);
   PrintFormat("Limites du chargement de l'historique à charger : début - %s, fin - %s
               TimeToString(start),TimeToString(end));
   InitCounters();
//---
   retourne(0);
  }
//+------------------------------------------------------------------+
//|  initialisation de la position, de l'ordre et du compteur        |
//+------------------------------------------------------------------+
void InitCounters()
  {
   ResetLastError();
//--- chargement de l'historique
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Echec du chargement de l'historique de %s à %s vers le cache. C
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- récupère la valeur courante
   orders=OrdersTotal();
   positions=PositionsTotal();
   deals=HistoryDealsTotal();
   history_orders=HistoryOrdersTotal();
   started=true;
   Print("Les compteurs des ordres, des positions et des transactions ont été initiali
  }
```

```
//+------------------------------------------------------------------+
//| Fonction de tick de l'Expert                                     |
//+------------------------------------------------------------------+
void OnTick()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| appelé lorsqu'un évènement Trade arrive                          |
//+------------------------------------------------------------------+
void OnTrade()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| ex de traitement des changements dans le trade et l'historique   |
//+------------------------------------------------------------------+
void SimpleTradeProcessor()
  {
   end=TimeCurrent();
   ResetLastError();
//--- téléchargement de l'historique de trading dans l'intervalle spécifié dans le cac
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Echec du chargement de l'historique de %s à %s vers le cache. C
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- récupère les valeurs courantes
   int curr_orders=OrdersTotal();
   int curr_positions=PositionsTotal();
   int curr_deals=HistoryDealsTotal();
   int curr_history_orders=HistoryOrdersTotal();
//--- vérifie si le nombre d'ordres actifs a été changé
   if(curr_orders!=orders)
     {
      //--- le nombre d'ordres actifs a changé
      PrintFormat("Le nombre d'ordres a changé. La valeur précédente était %d, la vale
                  orders,curr_orders);
      //--- mets à jour la valeur
      orders=curr_orders;
     }
//--- changements du nombre de positions ouvertes
   if(curr_positions!=positions)
     {
      //--- le nombre de positions ouvertes a changé
```

© 2000-2024, MetaQuotes Ltd.

```
      PrintFormat("Le nombre de positions a changé. La valeur précédente était %d, la
                  positions,curr_positions);
      //--- mets à jour la valeur
      positions=curr_positions;
     }
//--- changements du nombre de transactions dans le cache de l'historique des trades
   if(curr_deals!=deals)
     {
      //--- le nombre de transactions dans le cache de l'historique de trading a chang
      PrintFormat("Le nombre de transactions a changé. La valeur précédente était %d,
                  deals,curr_deals);
      //--- mets à jour la valeur
      deals=curr_deals;
     }
//--- changements dans le nombre d'ordres dans le cache de l'historique
   if(curr_history_orders!=history_orders)
     {
      //--- le nombre d'ordres dans le cache de l'historique a été changé
      PrintFormat("Le nombre d'ordres dans l'historique a été changé. La valeur précéd
                  history_orders,curr_history_orders);
     //--- mets à jour la valeur
     history_orders=curr_history_orders;
     }
//--- vérification s'il est nécessaire de changer les limites de l'historique de trade
   CheckStartDateInTradeHistory();
  }
//+------------------------------------------------------------------+
//|  changement de la date de début pour demander l'historique       |
//+------------------------------------------------------------------+
void CheckStartDateInTradeHistory()
  {
//--- intervalle initial, si nous commencions dès maintenant
   datetime curr_start=TimeCurrent()-days*PeriodSeconds(PERIOD_D1);
//--- on s'assure que la limite de début de l'historique de trading n'a pas disparu
//--- plus d'un jour par rapport à la date prévue
   if(curr_start-start>PeriodSeconds(PERIOD_D1))
     {
      //--- corrige la date de début de l'historique à charger dans le cache
      start=curr_start;
      PrintFormat("Nouvelle limite de début de l'historique à charger : début => %s",
                  TimeToString(start));
      //--- recharge maintenant l'historique de trading pour l'intervalle mis à jour
      HistorySelect(start,end);
      //--- corrige les compteurs de transactions et d'ordres dans l'historique pour l
      history_orders=HistoryOrdersTotal();
      deals=HistoryDealsTotal();
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
/* Exemple de sortie :
  Limites du chargement de l'historique à charger : début - 2018.07.16 18:11, fin - 20
  Les compteurs des ordres, des positions et des transactions ont été initialisés avec
  Le nombre d'ordres dans l'historique a été changé. Valeur précédente 0, valeur actue
  Le nombre d'ordres dans l'historique a été changé. Valeur précédente 1, valeur actue
  Le nombre de positions dans l'historique a été changé. Valeur précédente 0, valeur a
  Le nombre de transactions a changé. Valeur précédente 0, valeur actuelle 1
  Le nombre d'ordres dans l'historique a été changé. Valeur précédente 0, valeur actue
*/
```

Voir également

OrderSend, OrderSendAsync, OnTradeTransaction, Evènements du terminal client

© 2000-2024, MetaQuotes Ltd.

OnTradeTransaction

La fonction est appelée dans les EA lorsque l'évènement TradeTransaction apparaît. La fonction permet de traiter les résultats de l'exécution des demandes de trade.

```
void  OnTradeTransaction()
   const MqlTradeTransaction&    trans,     // structure de la transaction de trade
   const MqlTradeRequest&        request,   // structure de la demande
   const MqlTradeResult&         result     // structure de la réponse
   );
```

Parameters

trans

[in] variable de type MqlTradeTransaction décrivant une transaction effectuée sur un compte de trading.

request

[in] variable de type MqlTradeRequest décrivant une demande de trade qui a conduit à une transaction. Elle contient les valeurs pour la transaction de type TRADE_TRANSACTION_REQUEST uniquement.

result

[in] variable de type MqlTradeResult contenant le résultat de l'exécution d'une demande de trade qui a conduit à une transaction. Elle contient les valeurs pour la transaction de type TRADE_TRANSACTION_REQUEST uniquement.

Valeur de Retour

Aucune valeur de retour

Note

OnTradeTransaction() est appelé pour traiter l'évènement TradeTransaction envoyé par le serveur de trades au terminal dans les cas suivants :

- envoi d'une demande de trade depuis un programme MQL5 en utilisant les fonctions OrderSend()/OrderSendAsync() et leur exécution ultérieure ;
- envoi d'une demande de trade manuellement via l'interface utiliteur et son exécution ultérieure ;
- activations des ordres en attente et des ordres stop sur le serveur ;
- opérations sur le serveur de trades.

Les données sur le type transaction sont contenues dans le champ *type* de la variable *trans*. Les types de transactions de trades sont décrits dans l'énumération ENUM_TRADE_TRANSACTION_TYPE :

- TRADE_TRANSACTION_ORDER_ADD - ajout d'un nouvel ordre actif
- TRADE_TRANSACTION_ORDER_UPDATE - changement d'un ordre existant
- TRADE_TRANSACTION_ORDER_DELETE - suppression d'un ordre de la liste des ordres actifs
- TRADE_TRANSACTION_DEAL_ADD - ajout d'une transaction dans l'historique
- TRADE_TRANSACTION_DEAL_UPDATE - changement d'une transaction dans l'historique
- TRADE_TRANSACTION_DEAL_DELETE - suppression d'une transaction dans l'historique
- TRADE_TRANSACTION_HISTORY_ADD - ajout d'un ordre dans l'historique en résultat d'une exécution ou d'une annulation

© 2000-2024, MetaQuotes Ltd.

- TRADE_TRANSACTION_HISTORY_UPDATE – changement d'un ordre dans l'historique des ordres
- TRADE_TRANSACTION_HISTORY_DELETE – suppression d'un ordre dans l'historique des ordres
- TRADE_TRANSACTION_POSITION – changement d'une position non lié à l'exécution d'un trade
- TRADE_TRANSACTION_REQUEST – notification qu'une demande de trade a été traitée par le serveur et le résultat de son traitement a été reçu.

Lors du traitement des transactions de type TRADE_TRANSACTION_REQUEST, il est nécessaire d'analyser les 2ème et 3ème paramètres de la fonction OnTradeTransaction() – la *demande* et le *résultat* – pour récupérer des informations supplémentaires.

L'envoi d'une demande de trade entraîne une chaîne de transactions de trade sur un compte de trading : 1) la demande est acceptée pour traitement, 2) un ordre d'achat approprié est créé pour le compte, 3) l'ordre est ensuite exécuté, 4) l'ordre exécuté est supprimé de la liste des ordres actifs, 5) ajout à l'historique des ordres, 6) la transaction suivante est ajoutée à l'historique et 7) une nouvelle position est créée. Toutes ces étapes sont des transactions de trade. L'arrivée de chacune de ces transactions dans le terminal est l'évènement TradeTransaction. La priorité de l'arrivée de ces transactions dans le terminal n'est pas garantie. Vous ne devez donc pas vous attendre à ce qu'un groupe de transactions arrive l'un après l'autre lors du développement de votre algorithme de trading.

Lorsque les transactions sont traitées par la fonction OnTradeTransaction() de l'EA, le terminal continue à gérer les transactions de trade entrantes. Le status du compte de trading peut donc changer pendant l'exécution de OnTradeTransaction(). Par exemple, pendant qu'un programme MQL5 gère l'ajout d'un nouvel ordre, il peut être exécuté, supprimé de la liste des ordres ouverts et déplacé dans l'historique. Le programme est notifié de tous ces évènements.

La longueur de la queue de transactions est de 1024 éléments. Si OnTradeTransaction() traite déjà une autre transaction depuis longtemps, les précédentes peuvent être remplacées par de nouvelles transactions dans la queue.

La fonction OnTrade() est appelée après les appels à OnTradeTransaction() correspondants. En général, il n'y a pas de corrélation exacte entre le nombre d'appels à OnTrade() et à OnTradeTransaction(). Un appel à OnTrade() correspond à un ou plusieurs appels à OnTradeTransaction.

Chaque évènement Trade peut être le résultat d'une ou plusieurs demandes de trades. Les demandes de trades sont envoyées au serveur en utilisant OrderSend() ou OrderSendAsync(). Chaque demande peut aboutir à plusieurs évènements de trading. Vous ne pouvez pas vous fier à la phrase "Une demande - un évènement Trade", puisque le traitement des évènements peut être effectué en plusieurs étapes, et chaque opération peut changer l'état des ordres, des positions et de l'historique de trading.

EA exemple avec la fonction OnTradeTransaction()

```
//+------------------------------------------------------------------+
//|                                   OnTradeTransaction_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Exemple de listener des évènements TradeTransaction"
```

```
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   PrintFormat("DERNIER PING=%.f ms",
               TerminalInfoInteger(TERMINAL_PING_LAST)/1000.);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de tick de l'Expert                                     |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Fonction TradeTransaction                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//---
   static int counter=0;   // compteur d'appels à OnTradeTransaction()
   static uint lasttime=0; // heure du dernier appel à OnTradeTransaction()
//---
   uint time=GetTickCount();
//--- si la dernière transaction a été effectuée il y a plus d'1 seconde,
   if(time-lasttime>1000)
     {
      counter=0; // c'est donc une nouvelle opération de trade, et le compteur peut êt
      if(IS_DEBUG_MODE)
         Print(" Nouvelle opération de trade");
     }
   lasttime=time;
   counter++;
   Print(counter,". ",__FUNCTION__);
//--- résultat de l'exécution de la demande de trade
   ulong            lastOrderID   =trans.order;
   ENUM_ORDER_TYPE  lastOrderType =trans.order_type;
   ENUM_ORDER_STATE lastOrderState=trans.order_state;
//--- le nom du symbole pour lequel une transaction a été effectuée
   string trans_symbol=trans.symbol;
//--- type de transaction
   ENUM_TRADE_TRANSACTION_TYPE  trans_type=trans.type;
```

© 2000-2024, MetaQuotes Ltd.

```
   switch(trans.type)
     {
      case  TRADE_TRANSACTION_POSITION:   // modification de la position
        {
         ulong pos_ID=trans.position;
         PrintFormat("MqlTradeTransaction: Position  #%d %s modifiée: SL=%.5f TP=%.5f"
                     pos_ID,trans_symbol,trans.price_sl,trans.price_tp);
        }
      break;
      case TRADE_TRANSACTION_REQUEST:     // envoi d'une demande de trade
         PrintFormat("MqlTradeTransaction: TRADE_TRANSACTION_REQUEST");
         break;
      case TRADE_TRANSACTION_DEAL_ADD:    // ajout d'un trade
        {
         ulong           lastDealID   =trans.deal;
         ENUM_DEAL_TYPE lastDealType =trans.deal_type;
         double         lastDealVolume=trans.volume;
         //--- Identifiant du trade dans un système externe - un ticket assigné par un
         string Exchange_ticket="";
         if(HistoryDealSelect(lastDealID))
            Exchange_ticket=HistoryDealGetString(lastDealID,DEAL_EXTERNAL_ID);
         if(Exchange_ticket!="")
            Exchange_ticket=StringFormat("(Exchange deal=%s)",Exchange_ticket);

         PrintFormat("MqlTradeTransaction: %s deal #%d %s %s %.2f lot   %s",EnumToStri
                     lastDealID,EnumToString(lastDealType),trans_symbol,lastDealVolume
        }
      break;
      case TRADE_TRANSACTION_HISTORY_ADD: // ajout d'un ordre dans l'historique
        {
         //--- Identifiant de l'ordre dans un système externe - un ticket assigné par
         string Exchange_ticket="";
         if(lastOrderState==ORDER_STATE_FILLED)
           {
            if(HistoryOrderSelect(lastOrderID))
               Exchange_ticket=HistoryOrderGetString(lastOrderID,ORDER_EXTERNAL_ID);
            if(Exchange_ticket!="")
               Exchange_ticket=StringFormat("(Ticket boursier=%s)",Exchange_ticket);
           }
         PrintFormat("MqlTradeTransaction: %s ordre #%d %s %s %s   %s",EnumToString(tr
                     lastOrderID,EnumToString(lastOrderType),trans_symbol,EnumTostring
        }
      break;
      default: // autres transactions
        {
         //--- Identifiant de l'ordre dans un système externe - un ticket assigné par
         string Exchange_ticket="";
         if(lastOrderState==ORDER_STATE_PLACED)
           {
```

```
            if(OrderSelect(lastOrderID))
               Exchange_ticket=OrderGetString(ORDER_EXTERNAL_ID);
            if(Exchange_ticket!="")
               Exchange_ticket=StringFormat("(Ticket boursier=%s)",Exchange_ticket);
           }
         PrintFormat("MqlTradeTransaction: %s ordre #%d %s %s   %s",EnumToString(trans
                     lastOrderID,EnumToString(lastOrderType),EnumToString(lastOrderSta
        }
      break;
     }
//--- ticket de l'ordre
   ulong orderID_result=result.order;
   string retcode_result=GetRetcodeID(result.retcode);
   if(orderID_result!=0)
      PrintFormat("MqlTradeResult: ordre #%d retcode=%s ",orderID_result,retcode_resul
//---
  }
//+------------------------------------------------------------------+
//| convertit les codes de réponses en chaînes de caractères         |
//+------------------------------------------------------------------+
string GetRetcodeID(int retcode)
  {
   switch(retcode)
     {
      case 10004: return("TRADE_RETCODE_REQUOTE");             break;
      case 10006: return("TRADE_RETCODE_REJECT");              break;
      case 10007: return("TRADE_RETCODE_CANCEL");              break;
      case 10008: return("TRADE_RETCODE_PLACED");              break;
      case 10009: return("TRADE_RETCODE_DONE");                break;
      case 10010: return("TRADE_RETCODE_DONE_PARTIAL");        break;
      case 10011: return("TRADE_RETCODE_ERROR");               break;
      case 10012: return("TRADE_RETCODE_TIMEOUT");             break;
      case 10013: return("TRADE_RETCODE_INVALID");             break;
      case 10014: return("TRADE_RETCODE_INVALID_VOLUME");      break;
      case 10015: return("TRADE_RETCODE_INVALID_PRICE");       break;
      case 10016: return("TRADE_RETCODE_INVALID_STOPS");       break;
      case 10017: return("TRADE_RETCODE_TRADE_DISABLED");      break;
      case 10018: return("TRADE_RETCODE_MARKET_CLOSED");       break;
      case 10019: return("TRADE_RETCODE_NO_MONEY");            break;
      case 10020: return("TRADE_RETCODE_PRICE_CHANGED");       break;
      case 10021: return("TRADE_RETCODE_PRICE_OFF");           break;
      case 10022: return("TRADE_RETCODE_INVALID_EXPIRATION");  break;
      case 10023: return("TRADE_RETCODE_ORDER_CHANGED");       break;
      case 10024: return("TRADE_RETCODE_TOO_MANY_REQUESTS");   break;
      case 10025: return("TRADE_RETCODE_NO_CHANGES");          break;
      case 10026: return("TRADE_RETCODE_SERVER_DISABLES_AT");  break;
      case 10027: return("TRADE_RETCODE_CLIENT_DISABLES_AT");  break;
      case 10028: return("TRADE_RETCODE_LOCKED");              break;
      case 10029: return("TRADE_RETCODE_FROZEN");              break;
```

© 2000-2024, MetaQuotes Ltd.

```
      case 10030: return("TRADE_RETCODE_INVALID_FILL");        break;
      case 10031: return("TRADE_RETCODE_CONNECTION");          break;
      case 10032: return("TRADE_RETCODE_ONLY_REAL");           break;
      case 10033: return("TRADE_RETCODE_LIMIT_ORDERS");        break;
      case 10034: return("TRADE_RETCODE_LIMIT_VOLUME");        break;
      case 10035: return("TRADE_RETCODE_INVALID_ORDER");       break;
      case 10036: return("TRADE_RETCODE_POSITION_CLOSED");     break;
      default:
         return("TRADE_RETCODE_UNKNOWN="+IntegerToString(retcode));
         break;
     }
//---
  }
```

Voir également

OrderSend, OrderSendAsync, OnTradeTransaction, structure de demande de trade, Structure de transaction de trade, Types de transaction de trade, Types d'opération de trade, Evènements du terminal client

© 2000-2024, MetaQuotes Ltd.

OnBookEvent

La fonction est appelée dans les indicateurs et les EA lorsque l'évènement BookEvent apparaît. Il est prévu pour gérer les changements dans le Depth of Market.

```
void  OnBookEvent(
   const string&  symbol          // symbole
   );
```

Parameters

symbol

[in] Nom du symbole pour lequel le BookEvent est arrivé

Valeur de Retour

Aucune valeur de retour

Note

Pour récupérer les évènements BookEvent pour n'importe quel symbole, abonnez-vous simplement pour les recevoir pour ce symbole en utilisant la fonction MarketBookAdd(). Pour annuler un abonnement pour recevoir le BookEvent pour un symbole donné, appelez la fonction MarketBookRelease().

Le BookEvent est diffusé dans le graphique entier. Ceci signifie que si une application sur un graphique souscrit à BookEvent avec la fonction MarketBookAdd, tous les autres indicateurs et EA lancés sur le même graphique et ayant la fonction OnBookEvent() recevront également cet évènement. Il est donc nécessaire d'analyser le nom du symbole passé à la fonction OnBookEvent() dans le paramètre *symbol*.

Différents compteurs de BookEvent triés par symboles sont fournis pour toutes les applications en cours d'exécution sur un même graphique. Cela signifie que chaque graphique peut avoir plusieurs abonnements à différents symboles et un compteur est fourni pour chaque symbole. S'abonner et de désabonner aux changements BookEvent ne modifie le compteur d'abonnements que pour les symboles spécifiés dans un seul graphique. En d'autres termes, deux graphiques adjacents peuvent co-éxister pour le même symbole mais avec des valeurs différentes des compteurs d'abonnement.

La valeur initiale du compteur d'abonnement est zéro. A chaque appel à MarketBookAdd(), le compteur d'abonnement pour un symbole spécifié sur le graphique est augmenté de 1 (le symbole du graphique et le symbole de MarketBookAdd() n'ont pas besoin de correspondre). Lors de l'appel à MarketBookRelease(), le compteur d'abonnements pour un symbole spécifié dans le graphique est réduit de 1. Les évènements BookEvent pour n'importe quel symbole sont diffusés dans le graphique jusqu'à ce que le compteur soit égal à zéro. Il est donc important que chaque programme MQL5 contenant MarketBookAdd () appelle se désabonnent correctement des évènements pour chaque symbole en utilisant MarketBookRelease () à la fin de son travail. Pour cela, le nombre d'appels à MarketBookAdd() et à MarketBookRelease() doivent être égaux pour chaque appel pendant le cycle de vie entier du programme MQL5. L'utilisation de flags ou de compteurs d'abonnements personnalisés dans le programme vous permet de travailler en toute sécurité avec les événements BookEvent et empêche de désactiver les abonnements pour obtenir cet événement dans des programmes tiers dans le même graphique.

Les évènements BookEvent ne sont jamais ignorés et sont toujours placés dans une queue, même si le traitement du BookEvent précédent n'est pas encore terminé.

© 2000-2024, MetaQuotes Ltd.

Exemple

```
//+------------------------------------------------------------------+
//|                                          OnBookEvent_Sample.mq5 |
//|                       Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com/en/articles/2635"
#property version   "1.00"
#property description "Exemple de mesure du taux de rafraîchissement du market depth a
#property description "Le code est tiré de l'article https://www.mql5.com/en/articles/
//--- input parameters
input ulong ExtCollectTime   =30;  // heure du test en secondes
input ulong ExtSkipFirstTicks=10;  // nombre de ticks ignorés au démarrage
//--- flag d'abonnement aux évènements BookEvent
bool book_subscribed=false;
//--- tableau pour conserver les demandes du market depth
MqlBookInfo  book[];
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- affiche le démarrage
   Comment(StringFormat("En attente de l'arrivée des %I64u premiers ticks",ExtSkipFirs
   PrintFormat("En attente de l'arrivée des %I64u premiers ticks",ExtSkipFirstTicks);
//--- active la diffusion du market depth
   if(MarketBookAdd(_Symbol))
     {
      book_subscribed=true;
      PrintFormat("%s: la fonction MarketBookAdd(%s) a retourné true",__FUNCTION__,_Sy
     }
   else
      PrintFormat("%s: la fonction MarketBookAdd(%s) a retourné false ! GetLastError()
//--- initialisation réussie
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Déinitialise l'expert                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- affiche le code de la raison de la désinitialisation
   Print(__FUNCTION__,": Code de la raison de la désinitialisation = ",reason);
//--- annule l'abonnement pour récupérer les évènements du market depth
   if(book_subscribed)
     {
```

```
      if(!MarketBookRelease(_Symbol))
         PrintFormat("%s: MarketBookRelease(%s) a retourné false ! GetLastError()=%d",
      else
         book_subscribed=false;
     }
//---
  }
//+------------------------------------------------------------------+
//| Fonction BookEvent                                               |
//+------------------------------------------------------------------+
void OnBookEvent(const string &symbol)
  {
   static ulong starttime=0;              // heure de début du test
   static ulong tickcounter=0;            // compteur de mises à jour du market depth
//--- ne fonctionne avec les évènements du market depth que si nous nous y sommes abor
   if(!book_subscribed)
      return;
//--- mise à jour du compteur pour un certain symbole
   if(symbol!=_Symbol)
      return;
//--- ignore les premiers ticks pour effacer la queue et pour préparer
   tickcounter++;
   if(tickcounter<ExtSkipFirstTicks)
      return;
//--- conserve l'heure de début
   if(tickcounter==ExtSkipFirstTicks)
      starttime=GetMicrosecondCount();
//--- demande les données du market depth
   MarketBookGet(symbol,book);
//--- quand stopper ?
   ulong endtime=GetMicrosecondCount()-starttime;
   ulong ticks  =1+tickcounter-ExtSkipFirstTicks;
// combien de temps s'est-il écoulé en microsecondes depuis le début du test ?
   if(endtime>ExtCollectTime*1000*1000)
     {
      PrintFormat("%I64u ticks pour %.1f secondes: %.1f ticks/sec ",ticks,endtime/1000
      ExpertRemove();
      return;
     }
//--- affiche les compteurs dans le champ commentaire
   if(endtime>0)
      Comment(StringFormat("%I64u ticks pour %.1f secondes: %.1f ticks/sec ",ticks,end
  }
```

Voir également

MarketBookAdd, MarketBookRelease, MarketBookGet, OnTrade, OnTradeTransaction, OnTick, Fonctions de gestion des évènements, Exécution du programme, Evènements du terminal client

© 2000-2024, MetaQuotes Ltd.

OnChartEvent

La fonction est appelée dans les indicateurs et les EA lorsque l'évènement ChartEvent se produit. La fonction permet de gérer les changements du graphique opérés par l'utilisateur ou par un programme MQL5.

```
void  OnChartEvent()
   const int       id,       // identifiant de l'évènement
   const long&     lparam,   // paramètre de l'évènement de type long
   const double&   dparam,   // paramètre de l'évènement de type double
   const string&   sparam    // paramètre de l'évènement de type string
   );
```

Parameters

id

[in] Identifiant de l'évènement issu de l'énumération ENUM_CHART_EVENT.

lparam

[in] Paramètre de l'évènement de type long

dparam

[in] Paramètre de l'évènement de type double

sparam

[in] Paramètre de l'évènement de type string

Valeur de Retour

Aucune valeur de retour

Note

Il existe 11 types d'évènements pouvant être gérés avec la fonction prédéfinie OnChartEvent(). 65535 identifiants de CHARTEVENT_CUSTOM à CHARTEVENT_CUSTOM_LAST inclus sont fournis pour les évènements personnalisés. Pour générer un évènement personnalisé, utilisez la fonction EventChartCustom().

Descriptions courtes des évènements de l'énumération ENUM_CHART_EVENT :

- CHARTEVENT_KEYDOWN — appui d'une touche du clavier lorsque le focus est sur la fenêtre du graphique ;
- CHARTEVENT_MOUSE_MOVE — déplacement de la souris et clics sur l'un des boutons de la souris (si CHART_EVENT_MOUSE_MOVE=true pour le graphique) ;
- CHARTEVENT_OBJECT_CREATE — création d'un objet graphique (si CHART_EVENT_OBJECT_CREATE=true pour le graphique) ;
- CHARTEVENT_OBJECT_CHANGE — changement des propriétés d'un objet via la fenêtre des propriétés ;
- CHARTEVENT_OBJECT_DELETE — suppression d'un objet graphique (si CHART_EVENT_OBJECT_DELETE=true pour le graphique) ;
- CHARTEVENT_CLICK — clic sur le graphique ;
- CHARTEVENT_OBJECT_CLICK — clic de souris sur un objet graphique appartement au graphique ;
- CHARTEVENT_OBJECT_DRAG — déplacement d'un objet graphique avec la souris ;

© 2000-2024, MetaQuotes Ltd.

- CHARTEVENT_OBJECT_ENDEDIT — fin de l'édition d'un texte dans une zone de saisie d'un objet graphique (OBJ_EDIT) ;
- CHARTEVENT_CHART_CHANGE — changement de graphique ;
- CHARTEVENT_CUSTOM+n — identifiant d'un évènement personnalisé, où n est dans l'intervalle 0 à 65535. CHARTEVENT_CUSTOM_LAST contient le dernier identifiant possible pour un évènement personnalisé (CHARTEVENT_CUSTOM+65535).

Tous les programmes MQL5 fonctionnent dans des threads différents du thread principal de l'application. Le thread principal de l'application est responsable de la gestion de tous les messages systèmes de Windows et à son tour, génère des messages Windows pour sa propre utilisation. Par exemple, le déplacement de la souris sur le graphique (évènement WM_MOUSE_MOVE) génère plusieurs messages systèmes pour faire le rendu sur la fenêtre de l'application, et envoie aussi des messages internes aux experts et aux indicateurs lancés sur le graphique. Une situation peut se produire où le thread principal de l'application n'a pas encore traité le message système WM_PAINT (et n'a donc pas encore redessiné le graphique modifié), alors qu'un EA ou un indicateur a déjà reçu l'évènement du mouvement de la souris. Dans ce cas, la propriété CHART_FIRST_VISIBLE_BAR du graphique ne sera changée le rafraîchissement du graphique.

Pour chaque type d'évènement, les paramètres d'entrée de la fonction OnChartEvent() ont certaines valeurs nécessaires pour traiter cet évènement. Le tableau liste les évènements et les valeurs passées via les paramètres.

Evènement	Valeur du paramètre 'id'	Valeur du paramètre 'lparam'	Valeur du paramètre 'dparam'	Valeur du paramètre 'sparam'
Evènement d'appui sur une touche	CHARTEVENT_KEYDOWN	code de la touche pressée	Le nombre d'appuis générés pendant que la touche était maintenue pressée	Valeur sous forme d'une chaîne de caractères du masque de bits, qui décrit le statut des touches du clavier
Evènements souris (si CHART_EVENT_MOUSE_MOVE=true pour le graphique)	CHARTEVENT_MOUSE_MOVE	Coordonnée X	Coordonnée Y	Valeur sous forme d'une chaîne de caractères du masque de bits, qui décrit le statut des boutons de la souris
Evènement de la molette de la souris (si CHART_EVENT_MOUSE_WHEEL= true pour le graphique)	CHARTEVENT_MOUSE_WHEEL	Flags des états des touches et des boutons de la souris, les coordonnées X et Y du curseur. Voir la	La valeur Delta du défilement de la molette de la souris	—

© 2000-2024, MetaQuotes Ltd.

Evènement	Valeur du paramètre 'id'	Valeur du paramètre 'lparam'	Valeur du paramètre 'dparam'	Valeur du paramètre 'sparam'
		description dans l'exemple		
Création d'un objet graphique (si CHART_EVENT_OBJECT_CREATE =true pour le graphique)	CHARTEVENT_OBJECT_CREATE	—	—	Nom d'un objet graphique créé
Changement des propriétés d'un objet via la fenêtre des propriétés	CHARTEVENT_OBJECT_CHANGE	—	—	Nom d'un objet graphique modifié
Suppression d'un objet graphique (si CHART_EVENT_OBJECT_DELETE =true pour le graphique)	CHARTEVENT_OBJECT_DELETE	—	—	Nom d'un objet graphique supprimé
Clic de la souris sur un graphique	CHARTEVENT_CLICK	Coordonnée X	Coordonnée Y	—
Clic de la souris sur un objet graphique	CHARTEVENT_OBJECT_CLICK	Coordonnée X	Coordonnée Y	Nom d'un object graphique sur lequel l'évènement s'est produit
Déplacement d'un objet graphique avec la souris	CHARTEVENT_OBJECT_DRAG	—	—	Nom de l'objet graphique déplacé
Fin d'édition du texte dans la zone de saisie de l'objet graphique Champ d'Edition	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nom de l'objet graphique Champ d'Edition dans lequel l'édition du texte est terminée
Changement de la taille du graphique ou modification des	CHARTEVENT_CHART_CHANGE	—	—	—

© 2000-2024, MetaQuotes Ltd.

Evènement	Valeur du paramètre 'id'	Valeur du paramètre 'lparam'	Valeur du paramètre 'dparam'	Valeur du paramètre 'sparam'
propriétés du graphique via la boîte de dialogue des propriétés				
Evènement personnalisé avec le nombre N	CHARTEVENT_CUSTOM+N	Valeur définie par la fonction EventChartCustom()	Valeur définie par la fonction EventChartCustom()	Valeur définie par la fonction EventChartCustom()

Exemple de listener d'évènement du graphique :

```
//+------------------------------------------------------------------+
//|                                          OnChartEvent_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Exemple de listener d'évènement du graphique et générateur d'é
//--- identifiants des touches
#define KEY_NUMPAD_5       12
#define KEY_LEFT           37
#define KEY_UP             38
#define KEY_RIGHT          39
#define KEY_DOWN           40
#define KEY_NUMLOCK_DOWN   98
#define KEY_NUMLOCK_LEFT  100
#define KEY_NUMLOCK_5     101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP    104
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- affiche la valeur constante CHARTEVENT_CUSTOM
   Print("CHARTEVENT_CUSTOM=",CHARTEVENT_CUSTOM);
//---
   Print("Lancement de l'EA ",MQLInfoString(MQL5_PROGRAM_NAME));
//--- définit le flag pour recevoir les évènements de création des objets graphiques
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- définit le flag pour recevoir les évènements de suppression des objets graphique
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
```

```
//--- active les messages de défilement de la molette de la souris
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- la mise à jour forcée des propriétés du graphique permet de s'assurer de la prép
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de tick de l'Expert                                     |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- compteur de ticks pour générer un évènement personnalisé
   static int tick_counter=0;
//--- divise les ticks accumulés par cette valeur
   int simple_number=113;
//---
   tick_counter++;
//--- envoie un évènement personnalisé si le compteur de ticks est un multiple de simp
   if(tick_counter%simple_number==0)
     {
      //--- crée un identifiant d'évènement personnalisé de 0 à 65535
      ushort custom_event_id=ushort(tick_counter%65535);
      //---  envoie un évènement personnalisé
      EventChartCustom(ChartID(),custom_event_id,tick_counter,SymbolInfoDouble(Symbol
      //--- ajoute une ligne dans le journal pour analyser les résultats de l'exemple
      Print(__FUNCTION__,": évènement personnalisé envoyé, ID=",custom_event_id);
     }
//---
  }
//+------------------------------------------------------------------+
//| Fonction ChartEvent                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- appui sur une touche
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch((int)lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("Appui sur KEY_NUMLOCK_LEFT");   break;
         case KEY_LEFT:          Print("Appui sur KEY_LEFT");           break;
         case KEY_NUMLOCK_UP:    Print("Appui sur KEY_NUMLOCK_UP");     break;
         case KEY_UP:            Print("Appui sur KEY_UP");             break;
         case KEY_NUMLOCK_RIGHT: Print("Appui sur KEY_NUMLOCK_RIGHT");  break;
         case KEY_RIGHT:         Print("Appui sur KEY_RIGHT");          break;
```

© 2000-2024, MetaQuotes Ltd.

```
          case KEY_NUMLOCK_DOWN:  Print("Appui sur KEY_NUMLOCK_DOWN");   break;
          case KEY_DOWN:          Print("Appui sur KEY_DOWN");           break;
          case KEY_NUMPAD_5:      Print("Appui sur KEY_NUMPAD_5");       break;
          case KEY_NUMLOCK_5:     Print("Appui sur KEY_NUMLOCK_5");      break;
          default:                Print("Appui sur une touche non listée");
         }
      }
//--- clic du bouton gauche sur le graphique
   if(id==CHARTEVENT_CLICK)
      Print("Coordonnées du clic de la souris sur le graphique : x = ",lparam,"  y = '
//--- clic sur un objet graphique
   if(id==CHARTEVENT_OBJECT_CLICK)
      Print("Clic du bouton de la souris sur l'objet nommé '"+sparam+"'");
//--- objet supprimé
   if(id==CHARTEVENT_OBJECT_DELETE)
      Print("Suppression de l'objet nommé ",sparam);
//--- objet créé
   if(id==CHARTEVENT_OBJECT_CREATE)
      Print("Création de l'objet nommé ",sparam);
//--- modification de l'objet
   if(id==CHARTEVENT_OBJECT_CHANGE)
      Print("Changement de l'objet nommé ",sparam);
//--- objet déplacé ou changement des coordonnées du point d'ancrage
   if(id==CHARTEVENT_OBJECT_DRAG)
      Print("Changement des points d'ancrage de l'objet nommé ",sparam);
//--- changement du texte dans une zone de saisie de l'objet graphique Champ d'éditior
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      Print("Changement du texte dans l'objet Champ d'édition ",sparam,"  id=",id);
//--- Evènements des mouvements de la souris
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- Considère l'état des boutons et de la molette de la souris pour cet évèner
      int flg_keys = (int)(lparam>>32);          // Le flag de l'état des touches Ctrl
      int x_cursor = (int)(short)lparam;         // Coordonnée X où l'évènement de la
      int y_cursor = (int)(short)(lparam>>16);   // Coordonnée Y où l'évènement de la
      int delta    = (int)dparam;                // Valeur totale du défilement de la
      //--- gestion du flag
      string str_keys="";
      if((flg_keys&0x0001)!=0)
         str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0)
         str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0)
         str_keys+="SHIFT ";
      if((flg_keys&0x0008)!=0)
         str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0)
```

© 2000-2024, MetaQuotes Ltd.

```
         str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0)
         str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0)
         str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1)+"'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
     }
//--- Evènement du redimensionnement du graphique ou de la modification des propriétés
   if(id==CHARTEVENT_CHART_CHANGE)
      Print("Changement de la taille du graphique ou de ses propriétés");
//--- évènement personnalisé
   if(id>CHARTEVENT_CUSTOM)
      PrintFormat("Evènement personnalisé ID=%d, lparam=%d, dparam=%G, sparam=%s",id,l
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // souris gauche
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // souris droite
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // souris milieu
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // souris première bouton X
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // souris deuxième bouton X
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // touche shift
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // touche control
   return(res);
  }
```

Voir également

EventChartCustom, Types d'évènements graphiques, Fonctions de gestion des évènements, Exécution du programme, Evènements du terminal client

© 2000-2024, MetaQuotes Ltd.

OnTester

La fonction est appelée dans les Expert Advisors lorsque l'évènement Tester se produit pour effectuer les actions nécessaires après le test.

```
double  OnTester(void);
```

Valeur de Retour

Valeur du critère d'optimisation pour s'assurer des résultats du test.

Note

La fonction OnTester() ne peut être utilisée que lors du test d'EA et est prévue pour le calcul de la valeur utilisée comme critère 'Max personnalisé' lors de l'optimisation des paramètres d'entrée.

Pendant l'optimisation génétique, le tri des résultats dans une seule génération est effectué en tri descendant. Cela signifie que les résultats ayant la plus grande valeur sont réputés être les meilleurs du point de vue du critère d'optimisation. Les pires valeurs du tri sont placées à la fin et sont donc ignorées. Elles ne prennent donc pas part dans la formation de la génération suivante.

La fonction OnTester() vous permet donc non seulement de créer et de sauver vos rapports de résultats de test, mais aussi de contrôler le processus d'optimisation pour trouver les meilleurs paramètres de la stratégie de trading.

Ci-dessous se trouve un **exemple** de calcul du critère personnalisé d'optimisation. L'idée est de calculer la régression linéaire du graphique du solde. Elle est décrite dans l'article Optimiser une stratégie en utilisant le graphique du solde et en comparant les résultats du critère "Solde + max Sharpe Ratio".

```
//+------------------------------------------------------------------+
//|                                                OnTester_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "EA exemple avec la fonction OnTester()"
#property description "Comme critère d'optimisation personnalisé, "
#property description "le rapport de la régression linéaire du graphique du solde"
#property description "divisé par l'erreur quadratique moyenne est retourné"
//--- inclut la classe pour les opérations de trading
#include <Trade\Trade.mqh>
//--- paramètres d'entrée de l'EA
input double Lots               = 0.1;      // Volume
input int    Slippage           = 10;       // Slippage autorisé
input int    MovingPeriod       = 80;       // Période de la moyenne mobile
input int    MovingShift        = 6;        // Décalage de la moyenne mobile
//--- variables globales
int    IndicatorHandle=0;  // handle de l'indicateur
bool   IsHedging=false;    // flag pour le compte
CTrade trade;              // pour effectuer les opérations de trading
```

```
//---
#define EA_MAGIC 18052018
//+------------------------------------------------------------------+
//| Vérifie les conditions d'ouverture de position                   |
//+------------------------------------------------------------------+
void CheckForOpen(void)
  {
   MqlRates rt[2];
//--- ne trade qu'au début d'une nouvelle barre
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("Echec de CopyRates avec ",_Symbol,", aucun historique");
      return;
     }
//--- volume du tick
   if(rt[1].tick_volume>1)
      return;
//--- valeurs de la moyenne mobile
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("Echec de CopyBuffer de iMA, aucune données");
      return;
     }
//--- vérifie la présence d'un signal
   ENUM_ORDER_TYPE signal=WRONG_VALUE;
//--- la bougie a ouvert plus haut, mais a fermé en dessous de la moyenne mobile
   if(rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=ORDER_TYPE_BUY;    // signal d'achat
   else // bougie ouverte en dessous mais fermée au-dessus de la moyenne mobile
     {
      if(rt[0].open<ma[0] && rt[0].close>ma[0])
         signal=ORDER_TYPE_SELL;// signal de vente
     }
//--- vérifications supplémentaires
   if(signal!=WRONG_VALUE)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
        {
         double price=SymbolInfoDouble(_Symbol,signal==ORDER_TYPE_SELL ? SYMBOL_BID:SY
         trade.PositionOpen(_Symbol,signal,Lots,price,0,0);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Vérifie les conditions de fermeture de position                  |
//+------------------------------------------------------------------+
void CheckForClose(void)
```

```
  {
   MqlRates rt[2];
//--- ne trade qu'au début d'une nouvelle barre
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("Echec de CopyRates avec ",_Symbol,", aucun historique");
      return;
     }
   if(rt[1].tick_volume>1)
      return;
//--- valeurs de la moyenne mobile
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("Echec de CopyBuffer de iMA, aucune données");
      return;
     }
//--- la position a déjà été sélectionnée auparavant avec PositionSelect()
   bool signal=false;
   long type=PositionGetInteger(POSITION_TYPE);
//--- la bougie a ouvert plus haut, mais a fermé en dessous de la moyenne mobile - fe
   if(type==(long)POSITION_TYPE_SELL && rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=true;
//--- la bougie a ouvert plus bas, mais a fermé au-dessus de la moyenne mobile - ferme
   if(type==(long)POSITION_TYPE_BUY && rt[0].open<ma[0] && rt[0].close>ma[0])
      signal=true;
//--- vérifications supplémentaires
   if(signal)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
         trade.PositionClose(_Symbol,Slippage);
     }
//---
  }
//+------------------------------------------------------------------+
//| Sélectionne une position suivant le compte: Netting ou Hedging   |
//+------------------------------------------------------------------+
bool SelectPosition()
  {
   bool res=false;
//--- sélectionne une position pour un compte de type Hedging
   if(IsHedging)
     {
      uint total=PositionsTotal();
      for(uint i=0; i<total; i++)
        {
         string position_symbol=PositionGetSymbol(i);
         if(_Symbol==position_symbol && EA_MAGIC==PositionGetInteger(POSITION_MAGIC))
           {
```

```
            res=true;
            break;
           }
         }
      }
//--- sélectionne une position pour un compte de type Netting
   else
      {
       if(!PositionSelect(_Symbol))
          return(false);
       else
          return(PositionGetInteger(POSITION_MAGIC)==EA_MAGIC); //---vérifie le numéro
      }
//--- résultat de l'exécution
   return(res);
  }
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- définit le type de trading : Netting ou Hedging
   IsHedging=((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)==ACCOU
//--- initialise un objet pour le contrôle correct de la position
   trade.SetExpertMagicNumber(EA_MAGIC);
   trade.SetMarginMode();
   trade.SetTypeFillingBySymbol(Symbol());
   trade.SetDeviationInPoints(Slippage);
//--- crée l'indicateur de la moyenne mobile
   IndicatorHandle=iMA(_Symbol,_Period,MovingPeriod,MovingShift,MODE_SMA,PRICE_CLOSE);
   if(IndicatorHandle==INVALID_HANDLE)
      {
       printf("Erreur lors de la création de l'indicateur iMA");
       return(INIT_FAILED);
      }
//--- ok
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de tick de l'Expert                                     |
//+------------------------------------------------------------------+
void OnTick(void)
  {
//--- si une position est déjà ouverte, vérifie la condition de fermeture
   if(SelectPosition())
      CheckForClose();
// vérifie la position d'ouverture de position
   CheckForOpen();
//---
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Fonction de test                                                 |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- valeur d'optimisation du critère personnalisé
   double ret=0.0;
//--- récupère les résultats du trade dans le tableau
   double array[];
   double trades_volume;
   GetTradeResultsToArray(array,trades_volume);
   int trades=ArraySize(array);
//--- s'il y a moins de 10 trades, le test ne donne aucun résultat positif
   if(trades<10)
      return (0);
//--- résultat moyen par trade
   double average_pl=0;
   for(int i=0;i<ArraySize(array);i++)
      average_pl+=array[i];
   average_pl/=trades;
//--- affiche le message pour le mode de test simple
   if(MQLInfoInteger(MQL_TESTER) && !MQLInfoInteger(MQL_OPTIMIZATION))
      PrintFormat("%s: Trades=%d, Profit moyen=%.2f",__FUNCTION__,trades,average_pl);
//--- calcul des ratios de régression linéaire pour le graphique du profit
   double a,b,std_error;
   double chart[];
   if(!CalculateLinearRegression(array,chart,a,b))
      return (0);
//--- calcule l'erreur de la déviation du graphique depuis la ligne de régression
   if(!CalculateStdError(chart,a,b,std_error))
      return (0);
//--- calcule le ratio des bénéfices tendanciels sur l'écart type
   ret=(std_error == 0.0) ? a*trades : a*trades/std_error;
//--- retourne la valeur du critère personnalisé d'optimisation
   return(ret);
  }
//+------------------------------------------------------------------+
//| Récupère le tableau des profits/pertes des transactions          |
//+------------------------------------------------------------------+
bool GetTradeResultsToArray(double &pl_results[],double &volume)
  {
//--- demande l'historique de trading complet
   if(!HistorySelect(0,TimeCurrent()))
      return (false);
   uint total_deals=HistoryDealsTotal();
   volume=0;
//--- définit la taille initiale du tableau avec une marge - par le nombre de transact
   ArrayResize(pl_results,total_deals);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- compteur de transactions qui fixe le résultat de trading - perte ou profit
   int counter=0;
   ulong ticket_history_deal=0;
//--- parcours toutes les transactions
   for(uint i=0;i<total_deals;i++)
     {
      //--- sélectionne une transaction
      if((ticket_history_deal=HistoryDealGetTicket(i))>0)
        {
         ENUM_DEAL_ENTRY deal_entry  =(ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket_hi
         long            deal_type   =HistoryDealGetInteger(ticket_history_deal,DEAL_T
         double          deal_profit =HistoryDealGetDouble(ticket_history_deal,DEAL_PR
         double          deal_volume =HistoryDealGetDouble(ticket_history_deal,DEAL_VO
         //--- nous ne sommes intéressés que dans les opérations de trading
         if((deal_type!=DEAL_TYPE_BUY) && (deal_type!=DEAL_TYPE_SELL))
            continue;
         //--- seules les transactions fixent les pertes/profits
         if(deal_entry!=DEAL_ENTRY_IN)
           {
            //--- écrit le résultat de trading dans le tableau et augmente le compteur
            pl_results[counter]=deal_profit;
            volume+=deal_volume;
            counter++;
           }
        }
     }
//--- définit la taille finale du tableau
   ArrayResize(pl_results,counter);
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcule la régression linéaire y=a*x+b                           |
//+------------------------------------------------------------------+
bool CalculateLinearRegression(double  &change[],double &chartline[],
                               double  &a_coef,double  &b_coef)
  {
//--- vérifie la suffisance des données
   if(ArraySize(change)<3)
      return (false);
//--- crée un tableau du graphique avec une accumulation
   int N=ArraySize(change);
   ArrayResize(chartline,N);
   chartline[0]=change[0];
   for(int i=1;i<N;i++)
      chartline[i]=chartline[i-1]+change[i];
//--- calcule maintenant les ratios de régression
   double x=0,y=0,x2=0,xy=0;
   for(int i=0;i<N;i++)
     {
```

```
      x=x+i;
      y=y+chartline[i];
      xy=xy+i*chartline[i];
      x2=x2+i*i;
     }
   a_coef=(N*xy-x*y)/(N*x2-x*x);
   b_coef=(y-a_coef*x)/N;
//---
   return (true);
  }
//+------------------------------------------------------------------+
//| Calculer l'erreur d'écart quadratique moyen pour a et b spécifiés|
//+------------------------------------------------------------------+
bool  CalculateStdError(double  &data[],double  a_coef,double  b_coef,double &std_err)
  {
//--- somme des carrés d'erreur
   double error=0;
   int N=ArraySize(data);
   if(N<=2)
      return (false);
   for(int i=0;i<N;i++)
      error+=MathPow(a_coef*i+b_coef-data[i],2);
   std_err=MathSqrt(error/(N-2));
//---
   return (true);
  }
```

Voir également

Test de stratégies de trading, TesterHideIndicators, Travailler avec les résultats d'optimisation, TesterStatistics, OnTesterInit, OnTesterDeinit, OnTesterPass, MQL_TESTER, MQL_OPTIMIZATION, FileOpen, FileWrite, FileLoad, FileSave

© 2000-2024, MetaQuotes Ltd.

OnTesterInit

La fonction est appelée dans les EA lorsque l'évènement TesterInit se produit pour effectuer les actions nécessaires avant une optimisation dans le testeur de stratégies. Il y a 2 types de fonctions.

La **version qui retourne le résultat**

```
int  OnTesterInit(void);
```

Valeur de Retour

Une valeur de type int, zéro signifie l'initialisation réussie d'un EA lancé sur un graphique avant l'optimisation.

L'appel OnTesterInit() qui retourne le résultat de l'exécution est recommandé puisqu'il permet non seulement l'initialisation du programme, mais retourne également un code d'erreur dans le cas d'une fin prématurée de l'optimisation. Le retour de n'importe quelle valeur autre que INIT_SUCCEEDED (0) signifie une erreur, aucune optimisation n'est lancée.

La version sans retour de résultat n'est utilisé que pour la compatibilité avec les anciens codes. Il n'est pas recommandé de l'utiliser

```
void  OnTesterInit(void);
```

Note

L'évènement TesterInit est généré avant le lancement de l'optimisation de l'EA dans le testeur de stratégie. Au moment de cet évènement, un EA ayant les fonctions OnTesterDeInit() ou OnTesterPass() est téléchargé automatiquement sur un graphique séparé du terminal. Il a le symbole et la période qui ont été spécifiés dans le testeur.

Cet évènement reçoit les évènements TesterInit, TesterDeinit et TesterPass, mais pas Init, Deinit et NewTick ones. En conséquence, toute la logique nécessaire pour le traitement des résultats de chaque passe pendant l'optimisation devrait être implémentée dans les fonctions OnTesterInit (), OnTesterDeinit () et OnTesterPass ().

Le résultat de chaque passe individuelle pendant l'optimisation d'une stratégie peut être passé via un frame depuis la fonction OnTester () en utilisant la fonction FrameAdd ().

La fonction OnTesterInit() est utilisée pour initier un Expert Advisor avant le lancement de l'optimisation pour le traitement ultérieur des résultats de l'optimisation. Elle est toujours utilisée avec la fonction OnTesterDeinit().

Le temps d'exécution de OnTesterInit() est limité. S'il est trop grand, l'EA est stoppé obligatoirement, et l'optimisation est annulée. Un message est affiché dans le journal du testeur :

```
TesterOnTesterInit works too long. Le testeur ne peut pas être initialisé.
```

L'exemple est pris de OnTick. La fonction OnTesterInit() est ajoutée pour définir les paramètres de l'optimisation :

```
//+------------------------------------------------------------------+
//|                                          OnTesterInit_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "EA exemple avec la fonction OnTesterInit(),"
#property description "dans laquelle les valeurs et limites des "
#property description "entrées pendant l'optimisation sont définies"

input double lots=0. 1;       // volume en lots
input double kATR=3;          // longueur de la bougie de signal dans l'ATR
input int    ATRperiod=20;    // période de l'ATR
input int    holdbars=8;      // nombre de barre pour tenir la position
input int    slippage=10;     // slippage autorisé
input bool   revers=false;    // renversement de signal ?
input ulong  EXPERT_MAGIC=0;  // nombre magique de l'EA
//--- pour stocker le handle de l'ATR
int atr_handle;
//--- nous stockerons ici les dernières valeurs de l'ATR et le corps de la bougie
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//--- heure de lancement de l'optimisation
datetime optimization_start;
//--- pour l'affichage de la durée sur un graphique à la fin de l'optimisation
string report;
//+------------------------------------------------------------------+
//| Fonction TesterInit                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- set the values of inputs for optimization
   ParameterSetRange("lots",false,0.1,0,0,0);
   ParameterSetRange("kATR",true,3.0,1.0,0.3,7.0);
   ParameterSetRange("ATRperiod",true,10,15,1,30);
   ParameterSetRange("holdbars",true,5,3,1,15);
   ParameterSetRange("slippage",false,10,0,0,0);
   ParameterSetRange("revers",true,false,false,1,true);
   ParameterSetRange("EXPERT_MAGIC",false,123456,0,0,0);
   Print("Les valeurs initiales et les limites des paramètres d'optimisation sont déf
//--- heure de démarrage de l'optimisation
   optimization_start=TimeLocal();
   report=StringFormat("%s: optimisation lancée à %s",
                       __FUNCTION__,TimeToString(TimeLocal(),TIME_MINUTES|TIME_SECONDS
//--- affiche les messages sur le graphique et dans le journal du terminal
   Print(report);
   Comment(report);
//---
  }
//+------------------------------------------------------------------+
```

```
//| Fonction TesterDeinit                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- durée de l'optimisation
   string log_message=StringFormat("%s: l'optimisation a pris %d secondes",
                                   __FUNCTION__,TimeLocal()-optimization_start);
   PrintFormat(log_message);
   report=report+"\r\n"+log_message;
   Comment(report);
  }
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- initialisation des variables globales
   last_atr=0;
   last_body=0;
//--- définit le volume correct
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- crée le handle de l'indicateur ATR
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: échec de la création de iATR, code d'erreur %d",__FUNCTION__,Ge
      return(INIT_FAILED);
     }
//--- initialisation réussie de l'EA
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de tick de l'Expert                                     |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- signal de trading
   static int signal=0; // +1 signifie un signal d'achat, -1 signifie un signal de ver
//--- vérifie et ferme les anciennes positions ouvertes plus de 'holdbars' barres aupa
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- vérifie si c'est une nouvelle barre
   if(isNewBar())
     {
      //--- vérifie la présence d'un signal
      signal=CheckSignal();
     }
//--- si une position de type netting est ouverte, ignore le signal - attends jusqu'à
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       signal=0;
       return; // sors de la fonction NewTick et n'entre pas sur le marché avant qu'une
      }
//--- pour un compte de type hedging, chaque position est conservée et fermée séparéme
   if(signal!=0)
      {
       //--- signal d'achat
       if(signal>0)
         {
          PrintFormat("%s: Signal d'achat ! Revers=%s",__FUNCTION__,string(revers));
          if(Buy(trade_lot,slippage,EXPERT_MAGIC))
             signal=0;
         }
       //--- signal de vente
       if(signal<0)
         {
          PrintFormat("%s: Signal de vente ! Revers=%s",__FUNCTION__,string(revers));
          if(Sell(trade_lot,slippage,EXPERT_MAGIC))
             signal=0;
         }
      }
//--- fin de la fonction OnTick
  }
//+------------------------------------------------------------------+
//| Vérifie la présence d'un nouveau signal de trading               |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0 signifie aucun signal
   int res=0;
//--- récupère la valeur de l'ATR sur l'avant dernière barre (l'indice de la barre est
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
      {
       last_atr=atr_value[0];
       //--- récupère les données de la dernière barre fermée dans le tableau d'élément
       MqlRates bar[1];
       if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
         {
          //--- calcule la taille du corps de la dernière barre complétée
          last_body=bar[0].close-bar[0].open;
          //--- si le corps de la dernière barre (avec l'indice 1) est supérieur à la v
          if(MathAbs(last_body)>kATR*last_atr)
             res=last_body>0?1:-1; // valeur positive pour une bougie haussière
         }
       else
          PrintFormat("%s: Impossible de recevoir la dernière barre ! Erreur",__FUNCTIO
      }
```

© 2000-2024, MetaQuotes Ltd.

```
   else
      PrintFormat("%s: Impossible de recevoir la valeur de l'ATR ! Erreur",__FUNCTION_
//--- si le mode de trading renversé est activé
   res=revers?-res:res;  // retourne le signal si nécessaire (retourne -1 au lieu de 1
//--- retourne la valeur du signal de trading
   return (res);
  }
//+------------------------------------------------------------------+
//|  Retourne 'true' lorsqu'une nouvelle barre apparaît              |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // stocke l'heure d'ouverture de la barre courante
//--- récupère l'heure d'ouverture de la barre zéro
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- si l'heure d'ouverture change, une nouvelle barre est arrivée
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- affiche les données de l'heure d'ouverture de la nouvelle barre dans le jo
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- affiche un message avec l'heure d'ouverture de la nouvelle barre
         PrintFormat("%s: nouvelle barre sur %s %s ouverte à %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- récupère les données du dernier tick
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("Echec de SymbolInfoTick(), erreur = ",GetLastError());
         //--- affiche l'heure du dernier tick en millisecondes
         PrintFormat("Le dernier tick était à %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- nous avons une nouvelle barre
      return (true);
     }
//--- aucune nouvelle barre
   return (false);
  }
//+------------------------------------------------------------------+
//| Achète au prix du marché avec le volume spécifié                 |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- achète au prix du marché
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Vends au prix du marché avec le volume spécifié                  |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- vends au prix du marché
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Ferme les positions détenues trop longtemps (en barres)          |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // nombre de positions ouvertes
//--- itère sur les positions ouvertes
   for(int i=total-1; i>=0; i--)
     {
      //--- paramètres de la position
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      datetime position_open=(datetime)PositionGetInteger(POSITION_TIME);
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;

      //--- si la position est trop vieille et que le MagicNumber et le symbole corres
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
         double volume=PositionGetDouble(POSITION_VOLUME);
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // mets le texte en minuscules
         PrintFormat("Ferme la position #%d %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- définit le type de l'ordre et envoie une demande de trade
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Prépare et envoie une demande de trade                           |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
  {
//--- déclare et initialise les structures
   MqlTradeRequest request={};
```

© 2000-2024, MetaQuotes Ltd.

```
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- paramètres de la demande
   request.action   =TRADE_ACTION_DEAL;                     // trading operation type
   request.position =pos_ticket;                            // position ticket if clos
   request.symbol   =Symbol();                              // symbole
   request.volume   =volume;                                // volume
   request.type     =type;                                  // type de l'ordre
   request.price    =price;                                 // prix du trade
   request.deviation=slip;                                  // déviation du prix auto
   request.magic    =magicnumber;                           // MagicNumber de l'ordre
//--- envoie une demande
   if(!OrderSend(request,result))
     {
      //--- affiche les informations de l'échec
      PrintFormat("OrderSend %s %s %.2f à %.5f, erreur %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- information sur l'opération réussie
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Voir également

Test de stratégies de trading, Travailler avec les résultats d'optimisation, OnTesterDeinit, OnTesterPass, ParameterGetRange, ParameterSetRange

© 2000-2024, MetaQuotes Ltd.

OnTesterDeinit

La fonction est appelée dans les EA lorsque l'évènement TesterDeinit se produit après l'optimisation d'un EA.

```
void  OnTesterDeinit(void);
```

Valeur de Retour

Aucune valeur de retour

Note

L'évènement TesterDeinit est généré après la fin de l'optimisation de l'EA dans le testeur de stratégie.

Un EA ayant les fonctions OnTesterDeInit() ou OnTesterPass() est automatiquement téléchargé sur un graphique séparé au lancement de l'optimisation. Il a le symbole et la période qui ont été spécifiés dans le testeur. La fonction est prévue pour le traitement final de tous les résultats d'optimisation.

Gardez en tête que les frames d'optimisation envoyés par les agents de tests en utilisant la fonction FrameAdd() peuvent arriver en bloc et prennent du temps à être envoyés. Tous les frames, ainsi que les évènements TesterPass, peuvent donc arriver et être traités dans OnTesterPass() avant la fin de l'optimisation. Si vous souhaitez recevoir tous les frames liés dans OnTesterDeinit(), placez le bloc de code en utilisant la fonction FrameNext().

Voir également

Test de stratégies de trading, Travailler avec les résultats d'optimisation, TesterStatistics, OnTesterInit, OnTesterPass, ParameterGetRange, ParameterSetRange

© 2000-2024, MetaQuotes Ltd.

OnTesterPass

La fonction est appelée dans les EA lorsque l'évènement TesterPass se produit pour gérer l'arrivée d'un nouvel ensemble de données pendant l'optimisation d'un EA

```
void  OnTesterPass(void);
```

Valeur de Retour

Aucune valeur de retour

Note

L'évènement TesterPass est généré automatiquement lors de la réception d'un frame pendant l'optimisation d'un EA dans le testeur de stratégie.

Un EA ayant les fonctions OnTesterDeInit() ou OnTesterPass() est automatiquement téléchargé sur un graphique séparé au lancement de l'optimisation. Il a le symbole et la période qui ont été spécifiés dans le testeur. La fonction permet de gérer les frames reçus des agents de test pendant l'optimisation. Le frame contenant les résultats de test devrait être envoyé de la fonction OnTester() en utilisant la fonction FrameAdd().

Gardez en tête que les frames d'optimisation envoyés par les agents de tests en utilisant la fonction FrameAdd() peuvent arriver en bloc et prennent du temps à être envoyés. Tous les frames, ainsi que les évènements TesterPass, peuvent donc arriver et être traités dans OnTesterPass() avant la fin de l'optimisation. Si vous souhaitez recevoir tous les frames liés dans OnTesterDeinit(), placez le bloc de code en utilisant la fonction FrameNext().

A la fin de l'optimisation de OnTesterDeinit(), il est possible de trier à nouveau tous les frames reçus en utilisant les fonctions FrameFirst()/FrameFilter et FrameNext().

Voir également

Test de stratégies de trading, Travailler avec les résultats d'optimisation, OnTesterInit, OnTesterDeinit, FrameFirst, FrameFilter, FrameNext, FrameInputs

© 2000-2024, MetaQuotes Ltd.

La réception de l'information de marché

Les fonctions pour la réception de l'information sur l'état du marché.

Fonction	Action
SymbolsTotal	Rend le nombre accessible (choisi dans MarketWatch ou tous)des symboles
SymbolExist	Vérifie si un symbole avec le nom spécifié existe
SymbolName	Rend le nom du symbole indiqué
SymbolSelect	Choisit le symbole dans la fenêtre MarketWatch ou enlève le symbole de la fenêtre
SymbolIsSynchronized	Vérifie le synchronismedes données selon le symbole indiqué dans le terminal avec les données sur le serveur commercial
SymbolInfoDouble	Rend la valeur du type double du symbole indiqué pour la propriété correspondante
SymbolInfoInteger	Rend la valeur du type entier (long, datetime, int ou bool) du symbole indiqué pour la propriété correspondante
SymbolInfoString	Rend la valeur du type string du symbole indiqué pour la propriété correspondante
SymbolInfoMarginRate	Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера
SymbolInfoTick	Rend les valeurs courantes pour le symbole indiqué dans la variable du type MqlTick
SymbolInfoSessionQuote	Permet de recevoir le temps du début et le temps de la fin de la session de cotation indiquée pour le symbole et le jour de la semaine indiqués.
SymbolInfoSessionTrade	Permet de recevoir le temps du début et le temps de la fin de la session commerciale indiquée pour le symbole et le jour de la semaine indiqués.
MarketBookAdd	Assure l'ouverture du profondeur de marché selon l'instrument indiqué, ainsi que produit la souscription sur la réception des avis du changement du profondeur de marché indiqué
MarketBookRelease	Assure la clôture du profondeur de marché selon l'instrument, ainsi qui élimine la souscription sur la réception des avis du changement du profondeur de marché indiqué
MarketBookGet	Rend le tableau des structures du type MqlBookInfo, contenant les enregistrements du profondeur de marché du symbole indiqué

© 2000-2024, MetaQuotes Ltd.

SymbolsTotal

Rend le nombre accessible (choisi dans MarketWatch ou tous)des symboles.

```
int  SymbolsTotal(
   bool  selected      // true - seulement les symboles dans MarketWatch
   );
```

Paramètres

selected

[in] Le mode de la demande. Peut accepter les valeurs true ou false.

La valeur rendue

Si le paramètre selected est égal à true, revient le nombre de symboles choisis dans MarketWatch. Si la valeur false, revient le total de tous les symboles.

© 2000-2024, MetaQuotes Ltd.

SymbolExist

Vérifie si un symbole avec le nom spécifié existe.

```
bool  SymbolExist(
   const string  name,     // nom du symbole
   bool&   is_custom       // propriété de symbole personnalisé
   );
```

Parameters

name

[in] Nom du symbole.

is_custom

[out] Propriété de symbole personnalisé défini lors d'une exécution réussie. Si true, le symbole détecté est un symbole personnalisé.

Valeur de Retour

Si false, le symbole n'a pas été trouvé parmi les symboles standards et les symboles personnalisés.

Voir également

SymbolsTotal, SymbolSelect, Custom symbols

© 2000-2024, MetaQuotes Ltd.

SymbolName

Rend le nom du symbole indiqué.

```
string  SymbolName(
   int   pos,          // numéro dans la liste
   bool  selected      // true - seulement les symboles dans MarketWatch
   );
```

Paramètres

pos

[in] Le numéro du symbole par ordre.

selected

[in] Le mode de la demande. Si la valeur est true, on prend le symbole de la liste des choisis dans MarketWatch. Si la valeur est false, on prend le symbole de la liste commune.

La valeur rendue

La valeur du type string avec le nom du symbole.

© 2000-2024, MetaQuotes Ltd.

SymbolSelect

Choisit le symbole dans la fenêtre MarketWatch ou enlève le symbole de la fenêtre.

```
bool  SymbolSelect(
   string  name,        // nom du symbole
   bool    select       // brancher ou débrancher
   );
```

Paramètres

name

[in] Le nom du symbole.

select

[in] Le commutateur. Si la valeur est false, le symbole doit être enlevé de la fenêtre MarketWatch, dans le cas contraire le symbole doit être choisi à la fenêtre MarketWatch. Le symbole ne peut pas être enlevé, s'il y a des graphiques ouverts avec ce symbole ou il y a des positions ouvertes dans ce symbole.

La valeur rendue

En cas de l'échec la fonction rend false.

© 2000-2024, MetaQuotes Ltd.

SymbolIsSynchronized

Contrôle le fait de la synchronisation des données selon le symbole indiqué dans le terminal avec les données sur le serveur commercial

```
bool  SymbolIsSynchronized(
   string  name,        // nom du symbole
   );
```

Paramètres

name

[in] Le nom du symbole.

La valeur rendue

Si les données sont synchronisées, rend true, ou autrement rend false.

Voir aussi

SymbolInfoInteger, L'organisation de l'accès aux données

© 2000-2024, MetaQuotes Ltd.

SymbolInfoDouble

Rend la propriété correspondante du symbole indiqué. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
double  SymbolInfoDouble(
   string                    name,        // symbole
   ENUM_SYMBOL_INFO_DOUBLE   prop_id      // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  SymbolInfoDouble(
   string                    name,        // symbole
   ENUM_SYMBOL_INFO_DOUBLE   prop_id,     // identificateur de la propriété
   double&                   double_var   // acceptons par ici la valeur de la propriété
   );
```

Paramètres

name

[in] Le nom du symbole.

prop_id

[in] L'identificateur de la propriété du symbole. La valeur peut être une des valeurs ENUM_SYMBOL_INFO_DOUBLE.

double_var

[out] La variable du type double, qui prend la valeur de la propriété demandée.

La valeur rendue

La valeur du type double. En cas d'une mauvaise exécution on peut recevoir l'information sur l'erreurà l'aide de la fonctionGetLastError():

- 5040 - le paramètre de chaîne incorrecte pour l'indication du nom du symbole,
- 4301 - un symbole inconnu (l'instrument financier),
- 4302 - le symbole n'est pas choisi dans "l'Aperçu du marché" (est absent dans la liste des accessibles),
- 4303 - l'identificateur incorrecte de la propriété du symbole.

Note

Si la fonction est utilisée pour la réception de l'information sur un dernier tick, il vaut mieux utiliserSymbolInfoTick(). Il est possible que selon ce symbole dès le moment de la connexion du terminal au compte commercial il n'y avait pas encore aucune cotation. Dans un tel cas la valeur demandée sera incertaine.

Il suffit d'utiliser dans la plupart des cas la fonction SymbolInfoTick(), qui permet de recevoir pour un appel les valeurs Ask, Bid, Last, Volume et le temps de l'arrivée du dernier tick.

La fonction SymbolInfoMarginRate() fournit des données sur le montant de la marge facturée en fonction du type et de la direction de l'ordre.

© 2000-2024, MetaQuotes Ltd.

Exemple:

```
void OnTick()
  {
//--- recevrons le spread des propriétés du symbole
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d des points\r\n",
                            spreadfloat?"flottant":"fixé",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- maintenant calculerons nous-même le spread
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"spread calculé = "+(string)spread_points+" des points";
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

SymbolInfoInteger

Rend la propriété correspondante du symbole indiqué. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
long  SymbolInfoInteger(
   string                    name,       // symbole
   ENUM_SYMBOL_INFO_INTEGER  prop_id     // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  SymbolInfoInteger(
   string                    name,       // symbole
   ENUM_SYMBOL_INFO_INTEGER  prop_id,    // identificateur de la propriété
   long&                     long_var    // acceptons par ici la valeur de la propriété
   );
```

Paramètres

name

[in] Le nom du symbole.

prop_id

[in] L'identificateur de la propriété du symbole. La valeur peut être une des valeurs ENUM_SYMBOL_INFO_INTEGER.

long_var

[out] La valeur du type long, recevant la valeur de la propriété demandée.

La valeur rendue

La valeur du type long. En cas d'une mauvaise exécution on peut recevoir l'information sur l'erreurà l'aide de la fonctionGetLastError():

- 5040 - le paramètre de chaîne incorrecte pour l'indication du nom du symbole,
- 4301 - un symbole inconnu (l'instrument financier),
- 4302 - le symbole n'est pas choisi dans "l'Aperçu du marché" (est absent dans la liste des accessibles),
- 4303 - l'identificateur incorrecte de la propriété du symbole.

Note

Si la fonction est utilisée pour la réception de l'information sur un dernier tick, il vaut mieux utiliserSymbolInfoTick(). Il est possible que selon ce symbole dès le moment de la connexion du terminal au compte commercial il n'y avait pas encore aucune cotation. Dans un tel cas la valeur demandée sera incertaine.

Il suffit d'utiliser dans la plupart des cas la fonction SymbolInfoTick(), qui permet de recevoir pour un appel les valeurs Ask, Bid, Last, Volume et le temps de l'arrivée du dernier tick.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
void OnTick()
  {
//--- recevrons le spread des propriétés du symbole
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d des points\r\n",
                            spreadfloat?"flottant":"fixé",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- maintenant calculerons nous-même le spread
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"spread calculé = "+(string)spread_points+" des points";
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

SymbolInfoString

Rend la propriété correspondante du symbole indiqué. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
string  SymbolInfoString(
   string                    name,        // symbole
   ENUM_SYMBOL_INFO_STRING   prop_id      // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  SymbolInfoString(
   string                    name,        // symbole
   ENUM_SYMBOL_INFO_STRING   prop_id,     // identificateur de la propriété
   string&                   string_var   // acceptons par ici la valeur de la propriét
   );
```

Paramètres

name

[in] Le nom du symbole.

prop_id

[in] L'identificateur de la propriété du symbole. La valeur peut être une des valeurs ENUM_SYMBOL_INFO_STRING.

string_var

[out] La valeur du type string, recevant la valeur de la propriété demandée.

La valeur rendue

La valeur du type string. En cas d'une mauvaise exécution on peut recevoir l'information sur l'erreurà l'aide de la fonctionGetLastError():

- 5040 - le paramètre de chaîne incorrecte pour l'indication du nom du symbole,
- 4301 - un symbole inconnu (l'instrument financier),
- 4302 - le symbole n'est pas choisi dans "l'Aperçu du marché" (est absent dans la liste des accessibles),
- 4303 - l'identificateur incorrecte de la propriété du symbole.

Note

Si la fonction est utilisée pour la réception de l'information sur un dernier tick, il vaut mieux utiliserSymbolInfoTick(). Il est possible que selon ce symbole dès le moment de la connexion du terminal au compte commercial il n'y avait pas encore aucune cotation. Dans un tel cas la valeur demandée sera incertaine.

Il suffit d'utiliser dans la plupart des cas la fonction SymbolInfoTick(), qui permet de recevoir pour un appel les valeurs Ask, Bid, Last, Volume et le temps de l'arrivée du dernier tick.

© 2000-2024, MetaQuotes Ltd.

SymbolInfoMarginRate

Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера.

```
bool  SymbolInfoMarginRate(
   string              name,                      // символ
   ENUM_ORDER_TYPE     order_type,                // тип ордера
   double&             initial_margin_rate,       // коэффициент взимания начальной марж
   double&             maintenance_margin_rate    // коэффициент взимания поддерживающей
   );
```

Параметры

name

[in] Имя символа.

order_type

[in] Тип ордера.

initial_margin_rate

[in] Переменная типа double для получения коэффициента взимания начальной маржи. Начальная маржа - это размер гарантийной суммы под совершение сделки объемом в 1 лот соответствующего направления. Умножая коэффициент на начальную маржу, мы можем получить размер средств, который будет зарезервирован на счете при размещении ордера указанного типа.

maintenance_margin_rate

[out] Переменная типа double для получения коэффициента взимания поддерживающей маржи. Поддерживающая маржа - это размер минимальной суммы для поддержания открытой позиции объемом в 1 лот соответствующего направления. Умножая коэффициент на поддерживающую маржу, мы можем получить размер средств, который будет зарезервирован на счете после срабатывания ордера указанного типа.

Возвращаемое значение

Возвращает true в случае удачного выполнения запроса свойств, иначе false.

© 2000-2024, MetaQuotes Ltd.

SymbolInfoTick

Rend les prix courants pour le symbole indiqué dans la variable du type MqlTick.

```
bool  SymbolInfoTick(
   string    symbol,      // symbole
   MqlTick&  tick         // référence à la structure
   );
```

Paramètres

symbol

[in] Le nom du symbole.

tick

[out] Le lien à la structure du type MqlTick, où seront placés les prix courants et le temps de la dernière mise à jour des prix.

La valeur rendue

Rend true en cas du succès, autrement - false.

© 2000-2024, MetaQuotes Ltd.

SymbolInfoSessionQuote

Permet de recevoir le temps du début et le temps de la fin de la session de cotation indiquée pour le symbole et le jour de la semaine indiqués.

```
bool  SymbolInfoSessionQuote(
   string            name,                // le nom du symbole
   ENUM_DAY_OF_WEEK  day_of_week,         // le jour de la semaine
   uint              session_index,       // le numéro de la session
   datetime&         from,                // le temps du début de la session
   datetime&         to                   // le temps de la fin de la session
   );
```

Les paramètres

name

[in] Le nom du symbole.

ENUM_DAY_OF_WEEK

[in] Le jour de la semaine, la valeur de l'énumération ENUM_DAY_OF_WEEK.

uint

[in] Le numéro de série de la session, pour laquelle il est nécessaire de recevoir le temps du début et le temps de la fin. L'indexation des sessions commence par 0.

from

[out] Le temps du début de la session de 00 heures 00 minutes, il faut ignorer la date dans la valeur reçue.

to

[out] Le temps de la fin de la session de 00 heures 00 minutes, il faut ignorer la date dans la valeur reçue.

La valeur rendue

Si les données sont reçues pour les sessions indiquées, le symbole et le jour de la semaine, on rend true, autrement rend false.

Voir aussi

L'information sur l'instrument, TimeToStruct, La structure de la date

© 2000-2024, MetaQuotes Ltd.

SymbolInfoSessionTrade

Permet de recevoir le temps du début et le temps de la fin de la session commerciale indiquée pour le symbole et le jour de la semaine indiqués.

```
bool  SymbolInfoSessionTrade(
   string            name,                // le nom du symbole
   ENUM_DAY_OF_WEEK  day_of_week,         // le jour de la semaine
   uint              session_index,       // le numéro de la session
   datetime&         from,                // le temps du début de la session
   datetime&         to                   // le temps de la fin de la session
   );
```

Les paramètres

name

[in] Le nom du symbole.

ENUM_DAY_OF_WEEK

[in] Le jour de la semaine, la valeur de l'énumération ENUM_DAY_OF_WEEK.

uint

[in] Le numéro de série de la session, pour laquelle il est nécessaire de recevoir le temps du début et le temps de la fin. L'indexation des sessions commence par 0.

from

[out] Le temps du début de la session de 00 heures 00 minutes, il faut ignorer la date dans la valeur reçue.

to

[out] Le temps de la fin de la session de 00 heures 00 minutes, il faut ignorer la date dans la valeur reçue.

La valeur rendue

Si les données sont reçues pour les sessions indiquées, le symbole et le jour de la semaine, on rend true, autrement rend false.

Voir aussi

L'information sur l'instrument, TimeToStruct, La structure de la date

© 2000-2024, MetaQuotes Ltd.

MarketBookAdd

Assure l'ouverture du profondeur de marché selon l'instrument indiqué, ainsi que produit la souscription sur la réception des avis du changement du profondeur de marché indiqué.

```
bool  MarketBookAdd(
   string  symbol      // symbole
   );
```

Paramètres

symbol

[in] Le nom du symbole, dont le profondeur de marché est supposé d'utiliser dans l'expert donné ou le script.

La valeur rendue

La valeur true en cas de l'ouverture fructueuse, autrement false.

Note

Normalement, on doit appeler cette fonction de la fonction OnInit() ou dans le constructeur de la classe. Pour le traitement des avis venant dans le programme de l'expert doit être la fonction void OnBookEvent(string& symbol).

Voir aussi

La structure du profondeur de marché, Les structures et les classes

© 2000-2024, MetaQuotes Ltd.

MarketBookRelease

Assure la clôture du profondeur de marché selon l'instrument indiqué, ainsi qu'élimine la souscription sur la réception des avis du changement du profondeur de marché indiqué.

```
bool  MarketBookRelease(
   string  symbol      // nom du symbole
   );
```

Paramètres

symbol

[in] Le nom du symbole.

La valeur rendue

La valeur true en cas de la clôture fructueuse, autrement false.

Note

Normalement, on doit appeler cette fonction de la fonction OnDeinit() dans le cas où dans la fonction OnInit() a été appelée la fonction MarketBookAdd(). Ou elle doit appeler du destructeur de la classe, si dans le constructeur de cette classe s'appelle la fonction correspondante MarketBookAdd().

Voir aussi

La structure du profondeur de marché, Les structures et les classes

© 2000-2024, MetaQuotes Ltd.

MarketBookGet

Rend le tableau des structures du type MqlBookInfo, contenant les enregistrements du profondeur de marché du symbole indiqué.

```
bool  MarketBookGet(
   string        symbol,      // symbol
   MqlBookInfo&  book[]       // référence au tableau
   );
```

Paramètres

symbol

[in] Le nom du symbole.

book[]

[out] La référence au tableau des enregistrements du profondeur de marché. Le tableau peut être d'avance distribué pour le nombre suffisant des enregistrements. Si le tableau dynamique n'était pas d'avance distribué dans la mémoire d'exploitation, le terminal de client distribuera ce tableau lui-même.

La valeur rendue

Rend true en cas du succès, autrement - false.

Note

Le profondeur de marché doit être préalablement ouvert par la fonction MarketBookAdd().

Exemple:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo jusqu'au ",Symbol());
      for(int i=0;i<size;i++)
        {
         Print(i+": ",priceArray[i].price
               +"    Volume = "+priceArray[i].volume,
               " type = ",priceArray[i].type);
        }
     }
   else
     {
      Print("On n'a pas réussi à recevoir le contenu du profondeur de marché selon le
     }
```

Voir aussi

La structure du profondeur de marché, Les structures et les classes

© 2000-2024, MetaQuotes Ltd.

Fonctions du calendrier économique

Cette section décrit les fonctions utilisées pour travailler avec le calendrier économique disponible directement dans la plateforme MetaTrader. Le calendrier économique est une encyclopédie toute faite présentant des indicateurs macroéconomiques, leurs dates de publication et leur importance. Les valeurs pertinentes des indicateurs macroéconomiques sont envoyées à la plateforme MetaTrader dès le moment de la publication et sont affichées sur un graphique sous forme de balises vous permettant de suivre de manière visuelle les indicateurs requis par pays, par devise et par importance.

Toutes les fonctions permettant de travailler avec le calendrier économique utilisent l'heure du serveur de trading(TimeTradeServer). Cela signifie que l'heure dans la structure MqlCalendarValue et l'heure donnée dans les fonctions CalendarValueHistoryByEvent/CalendarValueHistory sont définies dans le fuseau horaire du serveur de trading, et non pas selon l'heure locale de l'utilisateur.

Les fonctions du calendrier économique permettent d'effectuer une analyse automatique des évènements selon leur propre critère d'importance pour les pays et devises correspondantes.

Fonction	Action
CalendarCountryById	Retourne la description d'un pays par son identifiant
CalendarEventById	Retourne la description d'un évènement par son identifiant
CalendarValueById	Retourne la description de la valeur d'un évènement par son identifiant
CalendarCountries	Retourne le tableau des noms des pays disponibles dans le calendrier
CalendarEventByCountry	Retourne le tableau des descriptions de tous les évènements disponibles dans le calendrier pour un code de pays spécifié
CalendarEventByCurrency	Retourne le tableau des descriptions de tous les évènements disponibles dans le calendrier pour une devise spécifiée
CalendarValueHistoryByEvent	Retourne le tableau des valeurs pour tous les évènements dans un intervalle de temps spécifié pour un identifiant d'évènement
CalendarValueHistory	Retourne le tableau des valeurs pour tous les évènements dans un intervalle de temps spécifié avec la possibilité de trier par pays et/ou devise
CalendarValueLastByEvent	Retourne le tableau des valeurs d'un évènement par son identifiant depuis le statut de la base de données du calendrier avec un change_id spécifié
CalendarValueLast	Retourne le tableau des valeurs de tous les évènements, avec la possibilité de trier par pays et/devise depuis le statut de la base de données du calendrier avec un change_id spécifié

© 2000-2024, MetaQuotes Ltd.

CalendarCountryById

Retourne la description d'un pays par son identifiant.

```
bool  CalendarCountryById(
   const long           country_id,      // identifiant du pays
   MqlCalendarCountry&  country          // variable pour récupérer la description du p
   );
```

Paramètres

country_id

[in] Identifiant du pays (ISO 3166-1).

country

[out] Variable de type MqlCalendarCountry pour retourner la description du pays.

Valeur de Retour

Retourne true en cas de succès, sinon retourne false. Pour obtenir des informations sur une erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 4001 - ERR_INTERNAL_ERROR (erreur générale d'exécution),
- 5402 - ERR_CALENDAR_NO_DATA (pays non trouvé),
- 5401 - ERR_CALENDAR_TIMEOUT (limite de temps de traitement de la requête atteinte).

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtient la liste des pays du calendrier économique
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- vérifie le résultat
   if(count==0)
      PrintFormat("CalendarCountries() a retourné 0 ! Error %d",GetLastError());
//--- s'il y a plus de 2 pays
   if(count>=2)
     {
      MqlCalendarCountry country;
      //--- obtient maintenant la description du pays par son identifiant
      if(CalendarCountryById(countries[1].id, country))
        {
         //--- prépare la description du pays
         string descr="id = "+IntegerToString(country.id)+"\n";
         descr+=("nom = " + country.name+"\n");
         descr+=("code = " + country.code+"\n");
         descr+=("devise = " + country.currency+"\n");
         descr+=("symbol_devise = " + country.currency_symbol+"\n");
```

© 2000-2024, MetaQuotes Ltd.

```
          descr+=("nom_url = " + country.url_name);
          //--- affiche la description du pays
          Print(descr);
         }
      else
          Print("Echec de CalendarCountryById(). Erreur ",GetLastError());
     }
//---
  }
/*
  Résultat :
   id = 999
   nom = European Union
   code = EU
  devise = EUR
  symbole_devise = €
   nom_url = european-union
*/
```

Voir aussi

CalendarCountries, CalendarEventByCountry

© 2000-2024, MetaQuotes Ltd.

CalendarEventById

Retourne la description d'un évènement par son identifiant.

```
bool  CalendarEventById(
   ulong                 event_id,      // identifiant de l'évènement
   MqlCalendarEvent&     event          // variable pour retourner la description d'un é
   );
```

Paramètres

event_id

[in] Identifiant de l'évènement.

event

[out] Variable de type MqlCalendarEvent pour retourner la description d'un évènement.

Valeur de Retour

Retourne true en cas de succès, sinon retourne false. Pour obtenir des informations sur une erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 4001 - ERR_INTERNAL_ERROR (erreur générale d'exécution),
- 5402 - ERR_CALENDAR_NO_DATA (pays non trouvé),
- 5401 - ERR_CALENDAR_TIMEOUT (limite de temps de traitement de la requête atteinte).

Note

Toutes les fonctions permettant de travailler avec le calendrier économique utilisent l'heure du serveur de trading(TimeTradeServer). Cela signifie que l'heure dans la structure MqlCalendarValue et l'heure donnée dans les fonctions CalendarValueHistoryByEvent/CalendarValueHistory sont définies dans le fuseau horaire du serveur de trading, et non pas selon l'heure locale de l'utilisateur.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- code du pays pour Germany (ISO 3166-1 Alpha-2)
   string germany_code="DE";
//--- récupère les évènements allemands
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(germany_code,events);
//--- affiche les évènements allemands dans le Journal
   if(events_count>0)
     {
      PrintFormat("Evènements allemands : %d",events_count);
      ArrayPrint(events);
     }
   else
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      PrintFormat("Echec de l'obtention des évènements pour le code de pays %s, erreur
                  germany_code,GetLastError());
      //--- fin prématurée du script
      return;
     }
//--- retourne la description du dernier évènements du tableau events[]
   MqlCalendarEvent event;
   ulong event_id=events[events_count-1].id;
   if(CalendarEventById(event_id,event))
     {
      MqlCalendarCountry country;
      CalendarCountryById(event.country_id,country);
      PrintFormat("Description de l'évènement avec event_id=%d reçu",event_id);
      PrintFormat("Pays : %s (code du pays = %d)",country.name,event.country_id);
      PrintFormat("Nom de l'évènement : %s",event.name);
      PrintFormat("Code de l'évènement : %s",event.event_code);
      PrintFormat("Importance de l'évènement : %s",EnumToString((ENUM_CALENDAR_EVENT_I
      PrintFormat("Type de l'évènement : %s",EnumToString((ENUM_CALENDAR_EVENT_TYPE)ev
      PrintFormat("Secteur de l'évènement : %s",EnumToString((ENUM_CALENDAR_EVENT_SECT
      PrintFormat("Fréquence de l'évènement : %s",EnumToString((ENUM_CALENDAR_EVENT_FR
      PrintFormat("Mode de publication de l'évènement : %s",EnumToString((ENUM_CALENDA
      PrintFormat("Unité de mesure de l'évènement : %s",EnumToString((ENUM_CALENDAR_EV
      PrintFormat("Nombre de décimales : %d",event.digits);
      PrintFormat("Multiplicateur de l'évènement : %s",EnumToString((ENUM_CALENDAR_EVE
      PrintFormat("URL source : %s",event.source_url);
     }
   else
      PrintFormat("Echec de l'obtention de la description de l'évènement pour event_d=
                  event_id,GetLastError());
  }
/*
  Résultat :
  Evènement allemands : 50
              [id] [type] [secteur][fréquence] [time_mode] [country_id] [unité][importa
   [ 0] 276010001      1        6           2           0          276      1
   [ 1] 276010002      1        6           2           0          276      1
   [ 2] 276010003      1        4           2           0          276      1
   [ 3] 276010004      1        4           2           0          276      1
   ....
   [47] 276500001      1        8           2           0          276      0
   [48] 276500002      1        8           2           0          276      0
   [49] 276500003      1        8           2           0          276      0
  Description de l'évènement avec event_id=276500003 reçu
  Pays : Allemagne (code du pays = 276)
  Nom de l'évènement : Markit Composite PMI
  Code de l'évènement : markit-composite-pmi
   Importance de l'évènement : CALENDAR_IMPORTANCE_MODERATE
   Type de l'évènement : CALENDAR_TYPE_INDICATOR
   Secteur de l'évènement : CALENDAR_SECTOR_BUSINESS
```

© 2000-2024, MetaQuotes Ltd.

```
    Fréquence de l'évènement : CALENDAR_FREQUENCY_MONTH
    Mode de publication de l'évènement : CALENDAR_TIMEMODE_DATETIME
    Unité de mesure de l'évènement : CALENDAR_UNIT_NONE
   Nombre de décimales : 1
    Multiplicateur de la valeur : CALENDAR_MULTIPLIER_NONE
    URL source : https://www.markiteconomics.com
*/
```

Voir aussi

CalendarEventByCountry, CalendarEventByCurrency, CalendarValueById

© 2000-2024, MetaQuotes Ltd.

CalendarValueById

Retourne la description de la valeur d'un évènement par son identifiant.

```
bool  CalendarValueById(
   ulong                value_id,      // identifiant de la valeur de l'évènement
   MqlCalendarValue&    value          // variable pour retourner la valeur de l'évène
   );
```

Paramètres

value_id

[in] Identifiant de la valeur de l'évènement.

value

[out] Variable de type MqlCalendarValue pour retourner la description de l'évènement. Voir l'exemple de gestion des événements du calendrier.

Valeur de Retour

Retourne true en cas de succès, sinon retourne false. Pour obtenir des informations sur une erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 4001 - ERR_INTERNAL_ERROR (erreur générale d'exécution),
- 5402 - ERR_CALENDAR_NO_DATA (pays non trouvé),
- 5401 - ERR_CALENDAR_TIMEOUT (limite de temps de traitement de la requête atteinte).

Note

La structure MqlCalendarValue fournit des méthodes pour vérifier et définir les valeurs des champs actual_value, forecast_value, prev_value et revised_prev_value. Si aucune valeur n'est spécifiée, le champ stocke LONG_MIN (-9223372036854775808).

Veuillez noter que les valeurs stockées dans ces champs sont multipliées par un million. Cela signifie que lorsque vous recevez des valeurs dans MqlCalendarValue à l'aide des fonctions CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent et CalendarValueLast, vous devez vérifier si les valeurs des champs sont égales à LONG_MIN ; si une valeur est spécifiée dans un champ, vous devez diviser la valeur par 1 000 000 afin d'obtenir la valeur. Une autre méthode pour obtenir les valeurs consiste à vérifier et à obtenir des valeurs en utilisant les fonctions de la structure MqlCalendarValue.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- code du pays pour le Japon (ISO 3166-1 Alpha-2)
   string japan_code="JP";
//--- définit les limites de l'intervalle duquel l'évènement est pris
   datetime date_from=D'01.01.2018';  // prends tous les évènements depuis 2018
   datetime date_to=0;                // 0 signifie tous les évènements connus, inclu
```

© 2000-2024, MetaQuotes Ltd.

```
//--- récupère toutes les valeurs des évènements du Japon
   MqlCalendarValue values[];
   int values_count=CalendarValueHistory(values,date_from,date_to,japan_code);
//--- déplacement parmi les valeurs détectées
   if(values_count>0)
     {
      PrintFormat("Nombre de valeurs pour les évènements du Japon : %d",values_count);
      //--- suppression de toutes les valeurs "vides" (actual_value==-922337203685477
      for(int i=values_count-1;i>=0;i--)
        {
         if(values[i].actual_value==-9223372036854775808)
            ArrayRemove(values,i,1);
        }
      PrintFormat("Nombre de valeurs après la suppression des valeurs vides : %d",Arra
     }
   else
     {
      PrintFormat("Echec de l'obtention des évènements pour le code de pays %s, erreur
                  japan_code,GetLastError());
      //--- fin prématurée du script
      return;
     }
//--- ne conserve que 10 valeurs dans le tableau values[]
   if(ArraySize(values)>10)
     {
      PrintFormat("Réduit la liste des valeurs à 10 et les affiche");
      ArrayRemove(values,0,ArraySize(values)-10);
     }
   ArrayPrint(values);

//--- affichons maintenant comment récupérer la description de la valeur d'un évènemen
   for(int i=0;i<ArraySize(values);i++)
     {
      MqlCalendarValue value;
      CalendarValueById(values[i].id,value);
      PrintFormat("%d : value_id=%d valeur=%d impact=%s",
                  i,values[i].id,value.actual_value,EnumToString(ENUM_CALENDAR_EVENT_I
     }
//---
  }
/*
  Résultat :
  Nombre de valeurs pour les évènements du Japon : 1734
  Nombre de valeurs après la suppression des valeurs vides : 1017
  Réduit la liste des valeurs à 10 et les affiche
        [id] [id_evenement]          [date_heure]      [période][révision] [valeur_act
   [0] 56500  392030004 2019.03.28 23:30:00 2019.03.01 00:00:00          0         90
   [1] 56501  392030005 2019.03.28 23:30:00 2019.03.01 00:00:00          0         70
   [2] 56502  392030006 2019.03.28 23:30:00 2019.03.01 00:00:00          0        110
```

© 2000-2024, MetaQuotes Ltd.

```
   [3] 56544  392030007 2019.03.28 23:30:00 2019.02.01 00:00:00          0        230(
   [4] 56556  392050002 2019.03.28 23:30:00 2019.02.01 00:00:00          0        163(
   [5] 55887  392020003 2019.03.28 23:50:00 2019.02.01 00:00:00          0         40(
   [6] 55888  392020004 2019.03.28 23:50:00 2019.02.01 00:00:00          0       -180(
   [7] 55889  392020002 2019.03.28 23:50:00 2019.02.01 00:00:00          0         20(
   [8] 55948  392020006 2019.03.28 23:50:00 2019.02.01 00:00:00          1        140(
   [9] 55949  392020007 2019.03.28 23:50:00 2019.02.01 00:00:00          1       -100(
  Affiche les données succintes des valeurs des évènements à partir de value_id
   0 : value_id=56500 valeur=900000 impact=CALENDAR_IMPACT_POSITIVE
   1 : value_id=56501 valeur=700000 impact=CALENDAR_IMPACT_NA
   2 : value_id=56502 valeur=1100000 impact=CALENDAR_IMPACT_POSITIVE
   3 : value_id=56544 valeur=2300000 impact=CALENDAR_IMPACT_NEGATIVE
   4 : value_id=56556 valeur=1630000 impact=CALENDAR_IMPACT_POSITIVE
   5 : value_id=55887 valeur=400000 impact=CALENDAR_IMPACT_NEGATIVE
   6 : value_id=55888 valeur=-1800000 impact=CALENDAR_IMPACT_POSITIVE
   7 : value_id=55889 valeur=200000 impact=CALENDAR_IMPACT_NEGATIVE
   8 : value_id=55948 valeur=1400000 impact=CALENDAR_IMPACT_POSITIVE
   9 : value_id=55949 valeur=-1000000 impact=CALENDAR_IMPACT_NEGATIVE
*/
```

Voir aussi

CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent, CalendarValueLast

© 2000-2024, MetaQuotes Ltd.

CalendarCountries

Retourne le tableau des noms des pays disponibles dans le Calendrier.

```
int  CalendarCountries(
   MqlCalendarCountry&  countries[]         // tableau pour récupérer la liste des des
   );
```

Paramètres

countries[]

[out] Un tableau d'éléments de type MqlCalendarCountry pour récupérer toutes les descriptions des pays du Calendrier.

Valeur de Retour

Nombre de descriptions retournées. Pour obtenir des informations sur une erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 4001 - ERR_INTERNAL_ERROR (erreur générale d'exécution),
- 5401 - ERR_CALENDAR_TIMEOUT (limite de temps de traitement de la requête atteinte),
- 5400 - ERR_CALENDAR_MORE_DATA (la taille du tableau est insuffisante pour recevoir les descriptions de tous les pays, toutes les descriptions ne seront pas retournées).

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtient la liste des pays du calendrier économique
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- affiche le tableau dans le Journal
   if(count>0)
      ArrayPrint(countries);
   else
      PrintFormat("CalendarCountries() a retourné 0 ! Error %d",GetLastError());
/*
  Résultat :
       [id]            [nom] [code] [devise] [symbole_devise]        [nom_url] [réservé]
   [ 0]     0 "Worldwide"       "WW"   "ALL"      ""                "worldwide"
   [ 1]   999 "European Union" "EU"   "EUR"      "€"               "european-union"
   [ 2]   840 "United States"  "US"   "USD"      "$"               "united-states"
   [ 3]   124 "Canada"         "CA"   "CAD"      "$"               "canada"
   [ 4]    36 "Australia"      "AU"   "AUD"      "$"               "australia"
   [ 5]   554 "New Zealand"    "NZ"   "NZD"      "$"               "new-zealand"
   [ 6]   392 "Japan"          "JP"   "JPY"      "Ī"               "japan"
   [ 7]   156 "China"          "CN"   "CNY"      "Ī"               "china"
   [ 8]   826 "United Kingdom" "GB"   "GBP"      "J"               "united-kingdom"
```

© 2000-2024, MetaQuotes Ltd.

```
   [ 9]  756 "Switzerland"    "CH"   "CHF"      "₣"                "switzerland"
   [10]  276 "Germany"        "DE"   "EUR"      "€"                "germany"
   [11]  250 "France"         "FR"   "EUR"      "€"                "france"
   [12]  380 "Italy"          "IT"   "EUR"      "€"                "italy"
   [13]  724 "Spain"          "ES"   "EUR"      "€"                "spain"
   [14]   76 "Brazil"         "BR"   "BRL"      "R$"               "brazil"
   [15]  410 "South Korea"    "KR"   "KRW"      "₩"                "south-korea"
*/
  }
```

Voir aussi

CalendarEventByCountry, CalendarCountryById

© 2000-2024, MetaQuotes Ltd.

CalendarEventByCountry

Retourne le tableau des descriptions de tous les évènements disponibles dans le Calendrier pour un code de pays spécifié.

```
int  CalendarEventByCountry(
   string                country_code,      // nom du code du pays (ISO 3166-1 alpha-2)
   MqlCalendarEvent&     events[]           // variable pour récupérer le tableau de des
   );
```

Paramètres

country_code

[in] Nom du code du pays (ISO 3166-1 alpha-2)

events[]

[out] Tableau de type MqlCalendarEvent pour retourner les descriptions de tous les évènements pour un pays spécifié.

Valeur de Retour

Nombre de descriptions retournées. Pour obtenir des informations sur une erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 4001 - ERR_INTERNAL_ERROR (erreur générale d'exécution),
- 4004 - ERR_NOT_ENOUGH_MEMORY (pas assez de mémoire pour exécuter une requête),
- 5401 - ERR_CALENDAR_TIMEOUT (limite de temps de traitement de la requête atteinte),
- erreurs d'exécutions de ArrayResize()

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- code du pays pour EU (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- récupère les évènements pour EU
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- affiche les évènements de EU dans le Journal
   if(events_count>0)
     {
      PrintFormat("Evènements pour EU : %d",events_count);
      ArrayPrint(events);
     }
//---
  }
/*
  Résultat :
  Evènements pour EU : 56
```

© 2000-2024, MetaQuotes Ltd.

```
              [id] [type]  [id_pays] [unité] [importance][multiplicateur] [chiffres] [e
   [ 0] 999010001      0        999      0            2             0         0  "ECB
   [ 1] 999010002      0        999      0            2             0         0  "ECB
   [ 2] 999010003      0        999      0            3             0         0  "ECB
   [ 3] 999010004      0        999      0            3             0         0  "ECB
   [ 4] 999010005      0        999      0            2             0         0  "ECB
   [ 5] 999010006      1        999      1            3             0         2  "ECB
   [ 6] 999010007      1        999      1            3             0         2  "ECB
   [ 7] 999010008      0        999      0            2             0         0  "ECB
   [ 8] 999010009      1        999      2            2             3         3  "ECB
   [ 9] 999010010      0        999      0            2             0         0  "ECB
   [10] 999010011      0        999      0            2             0         0  "ECB
   ...

*/
```

Voir aussi

CalendarCountries, CalendarCountryById

© 2000-2024, MetaQuotes Ltd.

CalendarEventByCurrency

Retourne le tableau des descriptions de tous les évènements disponibles dans le calendrier pour une devise spécifiée.

```
int  CalendarEventByCurrency(
   const string         currency,      // nom du code de la devise du pays
   MqlCalendarEvent&    events[]       // variable pour retourner le tableau de descrip
   );
```

Paramètres

currency

[in] Nom du code de la devise du pays.

events[]

[out] Tableau d'éléments de type MqlCalendarEvent pour retourner les descriptions de tous les évènements pour une devise spécifiée.

Valeur de Retour

Nombre de descriptions retournées. Pour obtenir des informations sur une erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 4001 - ERR_INTERNAL_ERROR (erreur générale d'exécution),
- 4004 - ERR_NOT_ENOUGH_MEMORY (pas assez de mémoire pour exécuter une requête),
- 5401 - ERR_CALENDAR_TIMEOUT (limite de temps de traitement de la requête atteinte),
- erreurs d'exécutions de ArrayResize()

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclaration du tableau pour récupérer les évèenements du calendrier économique
      MqlCalendarEvent events[];
//--- obtient les évènements de la devise EU
      int count = CalendarEventByCurrency("EUR",events);
      Print("quantité = ", count);
//--- 10 évènements sont suffisants pour cet exemple
      if(count>10)
         ArrayResize(events,10);
//--- affiche les évènements dans le Journal
      ArrayPrint(events);
  }
/*
  Résultat :
            [id] [type] [id_pays] [unité] [importance]
   [0] 999010001      0          999      0            2 "https://www.ecb.europa.eu/ho
```

© 2000-2024, MetaQuotes Ltd.

```
   [1] 999010002      0          999      0            2 "https://www.ecb.europa.eu/h
   [2] 999010003      0          999      0            3 "https://www.ecb.europa.eu/h
   [3] 999010004      0          999      0            3 "https://www.ecb.europa.eu/h
   [4] 999010005      0          999      0            2 "https://www.ecb.europa.eu/h
   [5] 999010006      1          999      1            3 "https://www.ecb.europa.eu/h
   [6] 999010007      1          999      1            3 "https://www.ecb.europa.eu/h
   [7] 999010008      0          999      0            2 "https://www.ecb.europa.eu/h
   [8] 999010009      1          999      2            2 "https://www.ecb.europa.eu/h
   [9] 999010010      0          999      0            2 "https://www.ecb.europa.eu/h
*/
```

Voir aussi

CalendarEventById, CalendarEventByCountry

© 2000-2024, MetaQuotes Ltd.

CalendarValueHistoryByEvent

Retourne le tableau des valeurs pour tous les évènements dans un intervalle de temps spécifié pour un identifiant d'évènement.

```
bool  CalendarValueHistoryByEvent(
   ulong              event_id,          // identifiant de l'évènement
   MqlCalendarValue&  values[],          // tableau des descriptions des valeurs
   datetime           datetime_from,     // bordure gauche de l'intervalle de temps
   datetime           datetime_to=0      // bordure droite de l'intervalle de temps
   );
```

Paramètres

event_id

[in] Identifiant de l'évènement.

values[]

[out] Tableau d'éléments de type MqlCalendarValue pour retourner les valeurs des évènements. Voir l'exemple de gestion des événements du calendrier.

datetime_from

[in] Date initiale de l'intervalle de temps à partir duquel les évènements sont sélectionnés pour un identifiant donné, avec *datetime_from* < *datetime_to*.

datetime_to=0

[in] Date de fin de l'intervalle de temps à partir duquel les évènements sont spécifiés pour un identifiant donné. Si *datetime_to* n'est pas donné (ou vaut 0), toutes les valeurs des évènements depuis la date *datetime_from* dans la base de données du Calendrier sont retournées (incluant les valeurs des évènements futurs).

Valeur de Retour

Retourne true en cas de succès, sinon retourne false. Pour obtenir des informations sur une erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 4001 - ERR_INTERNAL_ERROR (erreur générale d'exécution),
- 4004 - ERR_NOT_ENOUGH_MEMORY (pas assez de mémoire pour exécuter une requête),
- 5401 - ERR_CALENDAR_TIMEOUT (limite de temps de traitement de la requête atteinte),
- 5400 - ERR_CALENDAR_MORE_DATA (la taille du tableau est insuffisante pour recevoir les descriptions de toutes les valeurs, toutes les descriptions ne seront pas retournées).
- erreurs d'exécutions de ArrayResize()

Note

Toutes les fonctions permettant de travailler avec le calendrier économique utilisent l'heure du serveur de trading(TimeTradeServer). Cela signifie que l'heure dans la structure MqlCalendarValue et l'heure donnée dans les fonctions CalendarValueHistoryByEvent/CalendarValueHistory sont définies dans le fuseau horaire du serveur de trading, et non pas selon l'heure locale de l'utilisateur.

La structure MqlCalendarValue fournit des méthodes pour vérifier et définir les valeurs des champs actual_value, forecast_value, prev_value et revised_prev_value. Si aucune valeur n'est spécifiée, le champ stocke LONG_MIN (-9223372036854775808).

© 2000-2024, MetaQuotes Ltd.

Veuillez noter que les valeurs stockées dans ces champs sont multipliées par un million. Cela signifie que lorsque vous recevez des valeurs dans MqlCalendarValue à l'aide des fonctions CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent et CalendarValueLast, vous devez vérifier si les valeurs des champs sont égales à LONG_MIN ; si une valeur est spécifiée dans un champ, vous devez diviser la valeur par 1 000 000 afin d'obtenir la valeur. Une autre méthode pour obtenir les valeurs consiste à vérifier et à obtenir des valeurs en utilisant les fonctions de la structure MqlCalendarValue.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- code du pays pour EU (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- récupère les évènements pour EU
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- affiche les évènements de EU dans le Journal
   if(events_count>0)
     {
      PrintFormat("Evènements pour EU : %d",events_count);
      //--- réduit la liste des évènements, 10 évènements sont suffisants pour l'analy
      ArrayResize(events,10);
      ArrayPrint(events);
     }
//--- l'évènement "ECB Interest Rate Decision" a l'event_id=999010007
   ulong event_id=events[6].id;        // l'identifiant de l'évènement peut changer da
   string event_name=events[6].name;   // nom de l'évènement du Calendrier
   PrintFormat("Valeurs obtenues pour event_name=%s event_id=%d",event_name,event_id);
//--- récupère toutes les valeurs de l'évènement "ECB Interest Rate Decision"
   MqlCalendarValue values[];
//--- définit les limites de l'intervalle duquel l'évènement est pris
   datetime date_from=0;           // prends tous les évènements depuis le début de l'
   datetime date_to=D'01.01.2016'; // ne prends pas les évènements antérieurs à 2016
   if(CalendarValueHistoryByEvent(event_id,values,date_from,date_to))
     {
      PrintFormat("Valeurs reçues pour %s: %d",
                  event_name,ArraySize(values));
      //--- réduit la liste des valeurs, 10 valeurs sont suffisantes pour l'analyse
      ArrayResize(values,10);
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Erreur ! Echec de l'obtention des valeurs pour event_id=%d",event_i
      PrintFormat("Code d'erreur : %d",GetLastError());
     }
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//---
/*
  Résultat :
  Evènements pour EU : 56
            [id] [type] [secteur][fréquence] [time_mode] [country_id] [unité][importar
   [0] 999010001      0        5           0           0          999      0
   [1] 999010002      0        5           0           0          999      0
   [2] 999010003      0        5           0           0          999      0
   [3] 999010004      0        5           0           0          999      0
   [4] 999010005      0        5           0           0          999      0
   [5] 999010006      1        5           0           0          999      1
   [6] 999010007      1        5           0           0          999      1
   [7] 999010008      0        5           0           0          999      0
   [8] 999010009      1        5           0           0          999      2
   [9] 999010010      0        5           0           0          999      0
  Valeurs obtenues pour event_name=ECB Interest Rate Decision event_id=999010007
  Valeurs reçues pour ECB Interest Rate Decision: 102
       [id] [id_évènement]          [date_heure]      [periode][révision] [valeur_actu
   [0] 2776  999010007 2007.03.08 11:45:00 1970.01.01 00:00:00          0        3750
   [1] 2777  999010007 2007.05.10 11:45:00 1970.01.01 00:00:00          0        3750
   [2] 2778  999010007 2007.06.06 11:45:00 1970.01.01 00:00:00          0        4000
   [3] 2779  999010007 2007.07.05 11:45:00 1970.01.01 00:00:00          0        4000
   [4] 2780  999010007 2007.08.02 11:45:00 1970.01.01 00:00:00          0        4000
   [5] 2781  999010007 2007.09.06 11:45:00 1970.01.01 00:00:00          0        4000
   [6] 2782  999010007 2007.10.04 11:45:00 1970.01.01 00:00:00          0        4000
   [7] 2783  999010007 2007.11.08 12:45:00 1970.01.01 00:00:00          0        4000
   [8] 2784  999010007 2007.12.06 12:45:00 1970.01.01 00:00:00          0        4000
   [9] 2785  999010007 2008.01.10 12:45:00 1970.01.01 00:00:00          0        4000
*/
```

Voir aussi

CalendarCountries, CalendarEventByCountry, CalendarValueHistory, CalendarEventById, CalendarValueById

© 2000-2024, MetaQuotes Ltd.

CalendarValueHistory

Retourne le tableau des valeurs pour tous les évènements dans un intervalle de temps spécifié avec la possibilité de trier par pays et/ou devise.

```
bool  CalendarValueHistory(
   MqlCalendarValue&  values[],              // tableau pour les descriptions des vale
   datetime           datetime_from,         // bordure gauche de l'intervalle de temp
   datetime           datetime_to=0          // bordure droite de l'intervalle de temp
   const string       country_code=NULL,     // nom du code du pays (ISO 3166-1 alpha-
   const string       currency=NULL          // nom du code de la devise du pays
   );
```

Paramètres

values[]

[out] Tableau d'éléments de type MqlCalendarValue pour retourner les valeurs des évènements. Voir l'exemple de gestion des événements du calendrier.

datetime_from

[in] Date initiale de l'intervalle de temps à partir duquel les évènements sont sélectionnés pour un identifiant donné, avec *datetime_from* < *datetime_to*.

datetime_to=0

[in] Date de fin de l'intervalle de temps à partir duquel les évènements sont spécifiés pour un identifiant donné. Si *datetime_to* n'est pas donné (ou vaut 0), toutes les valeurs des évènements depuis la date *datetime_from* dans la base de données du Calendrier sont retournées (incluant les valeurs des évènements futurs).

country_code=NULL

[in] Nom du code du pays (ISO 3166-1 alpha-2)

currency=NULL

[in] Nom du code de la devise du pays.

Valeur de Retour

Retourne true en cas de succès, sinon retourne false. Pour obtenir des informations sur une erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 4001 - ERR_INTERNAL_ERROR (erreur générale d'exécution),
- 4004 - ERR_NOT_ENOUGH_MEMORY (pas assez de mémoire pour exécuter une requête),
- 5401 - ERR_CALENDAR_TIMEOUT (limite de temps de traitement de la requête atteinte),
- 5400 - ERR_CALENDAR_MORE_DATA (la taille du tableau est insuffisante pour recevoir les descriptions de toutes les valeurs, toutes les descriptions ne seront pas retournées).
- erreurs d'exécutions de ArrayResize()

Note

Toutes les fonctions permettant de travailler avec le calendrier économique utilisent l'heure du serveur de trading(TimeTradeServer). Cela signifie que l'heure dans la structure MqlCalendarValue et

© 2000-2024, MetaQuotes Ltd.

l'heure donnée dans les fonctions CalendarValueHistoryByEvent/CalendarValueHistory sont définies dans le fuseau horaire du serveur de trading, et non pas selon l'heure locale de l'utilisateur.

Si le tableau *events[]* de taille fixe passé à la fonction n'est pas assez grand, l'erreur ERR_CALENDAR_MORE_DATA (5400) est générée.

Si *datetime_to* n'est pas donné (ou vaut 0), toutes les valeurs des évènements depuis la date *datetime_from* dans la base de données du Calendrier sont retournées (incluant les valeurs des évènements futurs).

Pour les filtres *country_code* et *currency* , les valeurs NULL et "" sont équivalentes et signifie l'absence de filtre.

Pour *country_code*, le champ *code* de la structure MqlCalendarCountry, par exemple "US", "RU" ou "EU", doit être utilisé.

Pour *currency*, le champ *currency* de la structure MqlCalendarCountry, par exemple "USD", "RUB" ou "EUR", doit être utilisé.

Les filtres sont ajoutés les uns aux autres, c'est à dire que le 'ET' logique est utilisé pour ne sélectionner que les valeurs des évènements remplissant les 2 conditions (pays et devise).

La structure MqlCalendarValue fournit des méthodes pour vérifier et définir les valeurs des champs actual_value, forecast_value, prev_value et revised_prev_value. Si aucune valeur n'est spécifiée, le champ stocke LONG_MIN (-9223372036854775808).

Veuillez noter que les valeurs stockées dans ces champs sont multipliées par un million. Cela signifie que lorsque vous recevez des valeurs dans MqlCalendarValue à l'aide des fonctions CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent et CalendarValueLast, vous devez vérifier si les valeurs des champs sont égales à LONG_MIN ; si une valeur est spécifiée dans un champ, vous devez diviser la valeur par 1 000 000 afin d'obtenir la valeur. Une autre méthode pour obtenir les valeurs consiste à vérifier et à obtenir des valeurs en utilisant les fonctions de la structure MqlCalendarValue.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- code du pays pour EU (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- récupère toutes les valeurs des évènements pour EU
   MqlCalendarValue values[];
//--- définit les limites de l'intervalle duquel l'évènement est pris
   datetime date_from=D'01.01.2018';  // prends tous les évènements depuis 2018
   datetime date_to=0;                // 0 signifie tous les évènements connus, inclua
//--- demandes l'historique des évènements pour EU depuis 2018
   if(CalendarValueHistory(values,date_from,date_to,EU_code))
     {
      PrintFormat("Valeurs reçues des évènements pour country_code=%s: %d",
                  EU_code,ArraySize(values));
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- réduit la taille du tableau pour une sortie dans le Journal
      ArrayResize(values,10);
//--- affiche les valeurs des évènements dans le Journal
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Erreur ! Echec de l'obtention des évènements pour country_code=%s",
      PrintFormat("Code d'erreur : %d",GetLastError());
     }
//---
  }
/*
  Résultat :
  Valeurs reçues des évènements pour=EU: 1384
        [id] [id_evenement]       [date_heure]          [période][révision]   [valeur_a
   [0] 54215  999500001 2018.01.02 09:00:00 2017.12.01 00:00:00          3       60600
   [1] 54221  999500002 2018.01.04 09:00:00 2017.12.01 00:00:00          3       56600
   [2] 54222  999500003 2018.01.04 09:00:00 2017.12.01 00:00:00          3       58100
   [3] 45123  999030005 2018.01.05 10:00:00 2017.11.01 00:00:00          0         600
   [4] 45124  999030006 2018.01.05 10:00:00 2017.11.01 00:00:00          0        2800
   [5] 45125  999030012 2018.01.05 10:00:00 2017.12.01 00:00:00          1         900
   [6] 45126  999030013 2018.01.05 10:00:00 2017.12.01 00:00:00          1        1400
   [7] 54953  999520001 2018.01.05 20:30:00 2018.01.02 00:00:00          0      127900
   [8] 22230  999040003 2018.01.08 10:00:00 2017.12.01 00:00:00          0        9100
   [9] 22231  999040004 2018.01.08 10:00:00 2017.12.01 00:00:00          0       18400
*/
```

Voir aussi

CalendarCountries, CalendarEventByCountry, CalendarValueHistoryByEvent, CalendarEventById, CalendarValueById

© 2000-2024, MetaQuotes Ltd.

CalendarValueLastByEvent

Retourne le tableau des valeurs d'un évènement par son identifiant depuis le statut de la base de données du calendrier avec un change_id spécifié.

```
int  CalendarValueLastByEvent(
   ulong                 event_id,       // identifiant de l'évènement
   ulong&                change_id,      // identifiant de la valeur de l'évènement
   MqlCalendarValue&     values[]        // tableau des descriptions des valeurs
   );
```

Paramètres

event_id

[in] Identifiant de l'évènement.

change_id

[in][out] Identifiant du changement.

values[]

[out] Tableau d'éléments de type MqlCalendarValue pour retourner les valeurs des évènements. Voir l'exemple de gestion des événements du calendrier.

Valeur de Retour

Nombre de valeurs d'évènements retournées. Pour obtenir des informations sur une erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 4001 - ERR_INTERNAL_ERROR (erreur générale d'exécution),
- 4004 - ERR_NOT_ENOUGH_MEMORY (pas assez de mémoire pour exécuter une requête),
- 5401 - ERR_CALENDAR_TIMEOUT (limite de temps de traitement de la requête atteinte),
- 5400 - ERR_CALENDAR_MORE_DATA (la taille du tableau est insuffisante pour recevoir les descriptions de toutes les valeurs, toutes les descriptions ne seront pas retournées).
- erreurs d'exécutions de ArrayResize()

Note

Toutes les fonctions permettant de travailler avec le calendrier économique utilisent l'heure du serveur de trading(TimeTradeServer). Cela signifie que l'heure dans la structure MqlCalendarValue et l'heure donnée dans les fonctions CalendarValueHistoryByEvent/CalendarValueHistory sont définies dans le fuseau horaire du serveur de trading, et non pas selon l'heure locale de l'utilisateur.

Si le tableau *events[]* de taille fixe passé à la fonction n'est pas assez grand, l'erreur ERR_CALENDAR_MORE_DATA (5400) est générée.

Si *change_id* = 0 est passé à la fonction, la fonction retourne toujours 0 et la valeur de la base de données du calendrier courant est retournée dans *change_id*.

La fonction retourne le tableau des nouvelles spécifiées et un nouveau *change_id* qui peut être utilisé pour les appels ultérieurs à la fonction pour recevoir les nouvelles valeurs des évènements. Il est donc possible de mettre à jour les valeurs pour une nouvelle spécifiée en appelant cette fonction avec le dernier *change_id* connu.

© 2000-2024, MetaQuotes Ltd.

La structure MqlCalendarValue fournit des méthodes pour vérifier et définir les valeurs des champs actual_value, forecast_value, prev_value et revised_prev_value. Si aucune valeur n'est spécifiée, le champ stocke LONG_MIN (-9223372036854775808).

Veuillez noter que les valeurs stockées dans ces champs sont multipliées par un million. Cela signifie que lorsque vous recevez des valeurs dans MqlCalendarValue à l'aide des fonctions CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent et CalendarValueLast, vous devez vérifier si les valeurs des champs sont égales à LONG_MIN ; si une valeur est spécifiée dans un champ, vous devez diviser la valeur par 1 000 000 afin d'obtenir la valeur. Une autre méthode pour obtenir les valeurs consiste à vérifier et à obtenir des valeurs en utilisant les fonctions de la structure MqlCalendarValue.

L'échantillon EA écoutant la publication des rapports des Emplois Non Agricoles :

```
#property description "Exemple d'utilisation de la fonction CalendarValueLastByEvent"
#property description " pour suivre la publication du rapport des Emplois Non Agricole
#property description "Pour cela, récupère l'identifiant du changement courant"
#property description " de la base de données du Calendrier. Utilise ensuite cet ident
#property description " que les nouveaux évènements via le timer"
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le timer
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- détruit le timer
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Fonction tick de l'expert                                        |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Fonction Timer                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Identifiant de changement de la base de données du Calendrier
   static ulong calendar_change_id=0;
//--- attribut de 1er lancement
   static bool first=true;
//--- identifiant de l'évènement
   static ulong event_id=0;
//--- nom de l'évènement
   static string event_name=NULL;
//--- tableau de valeurs des évènements
   MqlCalendarValue values[];
//--- effecture l'initialisation - récupère le calendar_change_id courant
   if(first)
     {
      MqlCalendarEvent events[];
      //--- code du pays pour les USA (ISO 3166-1 Alpha-2)
      string USA_code="US";
      //--- récupère les évènements pour les USA
      int events_count=CalendarEventByCountry(USA_code,events);
      //--- position d'un évènement nécessaire dans le tableau 'events'
      int event_pos=-1;
      //--- affiche les évènements des USA dans le Journal
      if(events_count>0)
        {
         PrintFormat("%s: évènements de USA: %d",__FUNCTION__,events_count);
         for(int i=0;i<events_count;i++)
           {
            string event_name_low=events[i].name;
            //--- modifie le nom de l'évènement en minuscules
            if(!StringToLower(event_name_low))
              {
               PrintFormat("StringToLower() a retourné l'erreur %d ",GetLastError());
               //--- sors de la fonction rapidement
               return;
              }
            //--- cherche l'évènement "Nonfarm Payrolls"
            if(StringFind(event_name_low,"nonfarm payrolls")!=-1)
              {
               //--- évènement trouvé, conserve son identifiant
               event_id=events[i].id;
               //--- écrit le nom de l'évènement "Nonfarm Payrolls"
               event_name=events[i].name;
               //--- stocke la position de l'évènement dans le tableau 'events[]'
               event_pos=i;
               //--- rappelez-vous que le Calendrier comporte plusieurs événements dor
               PrintFormat("Evènement \"Nonfarm Payrolls\" trouvé: event_id=%d  event_
               //--- visualisez tous les événements en commentant l'opérateur 'break'
               break;
              }
           }
```

```
        //--- réduit la liste en supprimant les évènements après les "Nonfarm Payroll
        ArrayRemove(events,event_pos+1);
        //--- conserve 9 évènements avant "Nonfarm Payrolls" pour une meilleure analy
        ArrayRemove(events,0,event_pos-9);
        ArrayPrint(events);
      }
    else
      {
        PrintFormat("%s: CalendarEventByCountry(%s) a retourné 0 évènement, code d'er
                    USA_code,__FUNCTION__,GetLastError());
        //--- opération terminée en échec, réessaye au prochain appel du timer
        return;
      }

    //--- récupère l'identifiant de changement de la base de données du Calendrier p
    if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
      {
        //--- ce bloc de code ne peut pas être exécuté pendant le 1er lancement, mais
        PrintFormat("%s: Identifiant reçu courant de la base de données du Calendrier
                    __FUNCTION__,calendar_change_id);
        //--- définit le flag et sort avant le prochain évènement du timer
        first=false;
        return;
      }
    else
      {
        //--- les données ne sont pas reçus (normal au 1er lancement), vérifie s'il y
        int error_code=GetLastError();
        if(error_code==0)
          {
           PrintFormat("%s: Identifiant courant reçu de la base de données du Calendr
                       __FUNCTION__,calendar_change_id);
           //--- définit le flag et sort avant le prochain évènement du timer
           first=false;
           //--- nous avons maintenant la valeur de calendar_change_id
           return;
          }
        else
          {
           //--- c'est vraiment une erreur
           PrintFormat("%s: Echec de l'obtention des valeurs pour event_id=%d",__FUNC
           PrintFormat("Code d'erreur: %d",error_code);
        //--- opération terminée en échec, réessaye au prochain appel du timer
           return;
          }
      }
  }

//--- nous avons la dernière valeur connue de l'identifiant du changement du calendrie
```

© 2000-2024, MetaQuotes Ltd.

```
   ulong old_change_id=calendar_change_id;
//--- vérifie l'existence d'une nouvelle valeur pour l'évènement Nonfarm Payrolls
   if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
     {
      PrintFormat("%s: Nouveaux évèenements reçus pour \"%s\": %d",
                  __FUNCTION__,event_name,ArraySize(values));
      //--- affiche les données du tableau 'values' dans le Journal
      ArrayPrint(values);
      //--- affiche les valeurs des identifiants précédents et des nouveaux dans le Jo
      PrintFormat("%s: change_id précédent=%d, nouveau change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
/*
      écrivez ici votre code pour gérer la survenue d'évènements "Nonfarm Payrolls"
      */
     }
//---
  }
/*
  Résultat :
   OnTimer: Evènements de USA : 202
  Evènement "Nonfarm Payrolls" trouvé: event_id=840030016  event_name=Nonfarm Payrolls
            [id] [type] [secteur][fréquence] [time_mode] [country_id] [unité][importar
   [0] 840030007      1        4           2           0          840      1
   [1] 840030008      1        4           2           0          840      1
   [2] 840030009      1        4           2           0          840      0
   [3] 840030010      1        4           2           0          840      0
   [4] 840030011      1        4           2           0          840      1
   [5] 840030012      1        4           2           0          840      1
   [6] 840030013      1        4           2           0          840      1
   [7] 840030014      1        4           2           0          840      1
   [8] 840030015      1        3           2           0          840      1
   [9] 840030016      1        3           2           0          840      4
 OnTimer: Identifiant courant reçu de la base de données du Calendrier: change_id=3328

*/
```

Voir aussi

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

© 2000-2024, MetaQuotes Ltd.

CalendarValueLast

Retourne le tableau des valeurs de tous les évènements, avec la possibilité de trier par pays et/devise depuis le statut de la base de données du calendrier avec un change_id spécifié.

```
int  CalendarValueLast(
   ulong&              change_id,            // identifiant de la valeur de l'évèner
   MqlCalendarValue&   values[],             // tableau des descriptions des valeurs
   const string        country_code=NULL,    // nom du code du pays (ISO 3166-1 alph
   const string        currency=NULL         // nom du code de la devise du pays
   );
```

Paramètres

change_id

[in][out] Identifiant du changement.

values[]

[out] Tableau d'éléments de type MqlCalendarValue pour retourner les valeurs des évènements. Voir l'exemple de gestion des événements du calendrier.

country_code=NULL

[in] Nom du code du pays (ISO 3166-1 alpha-2)

currency=NULL

[in] Nom du code de la devise du pays.

Valeur de Retour

Nombre de valeurs d'évènements retournées. Pour obtenir des informations sur une erreur, appelez la fonction GetLastError(). Erreurs possibles :

- 4001 - ERR_INTERNAL_ERROR (erreur générale d'exécution),
- 4004 - ERR_NOT_ENOUGH_MEMORY (pas assez de mémoire pour exécuter une requête),
- 5401 - ERR_CALENDAR_TIMEOUT (limite de temps de traitement de la requête atteinte),
- 5400 - ERR_CALENDAR_MORE_DATA (la taille du tableau est insuffisante pour recevoir les descriptions de toutes les valeurs, toutes les descriptions ne seront pas retournées).
- erreurs d'exécutions de ArrayResize()

Note

Toutes les fonctions permettant de travailler avec le calendrier économique utilisent l'heure du serveur de trading(TimeTradeServer). Cela signifie que l'heure dans la structure MqlCalendarValue et l'heure donnée dans les fonctions CalendarValueHistoryByEvent/CalendarValueHistory sont définies dans le fuseau horaire du serveur de trading, et non pas selon l'heure locale de l'utilisateur.

Si le tableau *events[]* de taille fixe passé à la fonction n'est pas assez grand, l'erreur ERR_CALENDAR_MORE_DATA (5400) est générée.

Si *change_id* = 0 est passé à la fonction, la fonction retourne toujours 0 et la valeur de la base de données du calendrier courant est retournée dans *change_id*.

Pour les filtres *country_code* et *currency* , les valeurs NULL et "" sont équivalentes et signifie l'absence de filtre.

© 2000-2024, MetaQuotes Ltd.

Pour *country_code*, le champ *code* de la structure MqlCalendarCountry, par exemple "US", "RU" ou "EU", doit être utilisé.

Pour *currency*, le champ *currency* de la structure MqlCalendarCountry, par exemple "USD", "RUB" ou "EUR", doit être utilisé.

Les filtres sont ajoutés les uns aux autres, c'est à dire que le 'ET' logique est utilisé pour ne sélectionner que les valeurs des évènements remplissant les 2 conditions (pays et devise).

La fonction retourne le tableau des nouvelles spécifiées et un nouveau *change_id* qui peut être utilisé pour les appels ultérieurs à la fonction pour recevoir les nouvelles valeurs des évènements. Il est donc possible de mettre à jour les valeurs pour une nouvelle spécifiée en appelant cette fonction avec le dernier *change_id* connu.

La structure MqlCalendarValue fournit des méthodes pour vérifier et définir les valeurs des champs actual_value, forecast_value, prev_value et revised_prev_value. Si aucune valeur n'est spécifiée, le champ stocke LONG_MIN (-9223372036854775808).

Veuillez noter que les valeurs stockées dans ces champs sont multipliées par un million. Cela signifie que lorsque vous recevez des valeurs dans MqlCalendarValue à l'aide des fonctions CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent et CalendarValueLast, vous devez vérifier si les valeurs des champs sont égales à LONG_MIN ; si une valeur est spécifiée dans un champ, vous devez diviser la valeur par 1 000 000 afin d'obtenir la valeur. Une autre méthode pour obtenir les valeurs consiste à vérifier et à obtenir des valeurs en utilisant les fonctions de la structure MqlCalendarValue.

L'échantillon EA écoutant les événements du calendrier économique :

```
#property description "Exemple d'utilisation de la fonction CalendarValueLast"
#property description " pour développer le listener des évènements du calendrier écono
#property description "Pour cela, récupère l'identifiant du changement courant"
#property description " de la base de données du Calendrier. Utilise ensuite cet ident
#property description " que les nouveaux évènements via le timer"
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le timer
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- détruit le timer
   EventKillTimer();
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Fonction tick de l'expert                                        |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Fonction Timer                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- Identifiant de changement de la base de données du Calendrier
   static ulong calendar_change_id=0;
//--- attribut de 1er lancement
   static bool first=true;
//--- tableau de valeurs des évènements
   MqlCalendarValue values[];
//--- effecture l'initialisation - récupère le calendar_change_id courant
   if(first)
     {
      //--- récupère l'identifiant de changement de la base de données du Calendrier
      if(CalendarValueLast(calendar_change_id,values)>0)
        {
         //--- ce bloc de code ne peut pas être exécuté pendant le 1er lancement, mais
         PrintFormat("%s: Identifiant reçu courant de la base de données du Calendrier
                     __FUNCTION__,calendar_change_id);
         //--- définit le flag et sort avant le prochain évènement du timer
         first=false;
         return;
        }
      else
        {
         //--- les données ne sont pas reçus (normal au 1er lancement), vérifie s'il y
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: Identifiant courant reçu de la base de données du Calendr
                        __FUNCTION__,calendar_change_id);
            //--- définit le flag et sort avant le prochain évènement du timer
            first=false;
            //--- nous avons maintenant la valeur de calendar_change_id
            return;
           }
         else
           {
            //--- c'est vraiment une erreur
            PrintFormat("%s: Echec de l'obtention des évènements dans CalendarValueLas
                        __FUNCTION__,error_code);
```

© 2000-2024, MetaQuotes Ltd.

```
            //--- opération terminée en erreur, réinitialisation lors du prochain appe
            return;
           }
        }
     }

//--- nous avons la dernière valeur connue de l'identifiant du changement du calendrie
   ulong old_change_id=calendar_change_id;
//--- vérifie s'il y a de nouveaux évènements dans le Calendrier
   if(CalendarValueLast(calendar_change_id,values)>0)
     {
      PrintFormat("%s: Nouveaux évènements reçus du Calendrier: %d",
                  __FUNCTION__,ArraySize(values));
      //--- affiche les données du tableau 'values' dans le Journal
      ArrayPrint(values);
      //--- affiche les valeurs des identifiants précédents et des nouveaux dans le Jo
      PrintFormat("%s: change_id précédent=%d, nouveau change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
      //--- affiche les nouveaux évènements dans le Journal
      ArrayPrint(values);
      /*
     écrivez ici votre code pour gérer la survenue d'évènements
      */
     }
//---
  }
/*
  Exemple d'utilisation du listener :
  OnTimer: Identifiant courant reçu de la base de données du Calendrier: change_id=332
  OnTimer: Nouveaux évènements reçus du Calendrier: 1
       [id] [id_évènement]          [date_heure]      [période][révision] [valeur_actu
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -5077
  OnTimer: change_id précédent=33281792, nouveau change_id=33282048
       [id] [id_évènement]          [date_heure]      [période][révision] [valeur_actu
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -5077
  OnTimer: Nouveaux évènements reçus du Calendrier: 1
       [id] [id_évènement]          [date_heure]      [période][révision]       [valeu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -922337203
  OnTimer: change_id précédent=33282048, nouveau change_id=33282560
       [id] [id_évènement]          [date_heure]      [période][révision]       [valeu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -922337203

*/
```

Voir aussi

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Accès aux Timeseries et aux Données d'un Indicateur

Ce sont les fonctions permettant d'utiliser les timeseries et les indicateurs. Une série chronologique, ou timeserie, diffère des données habituelles par son ordre inverse - les éléments des timeseries sont indexés de la fin du tableau vers son début (des données les plus récentes aux plus anciennes). Pour copier les valeurs des séries chronologiques et les données d'un indicateur, il est recommandé de n'utiliser que des tableaux dynamiques, car les fonctions de copie sont conçues pour n'allouer que la taille nécessaire aux tableaux pour recevoir les valeurs.

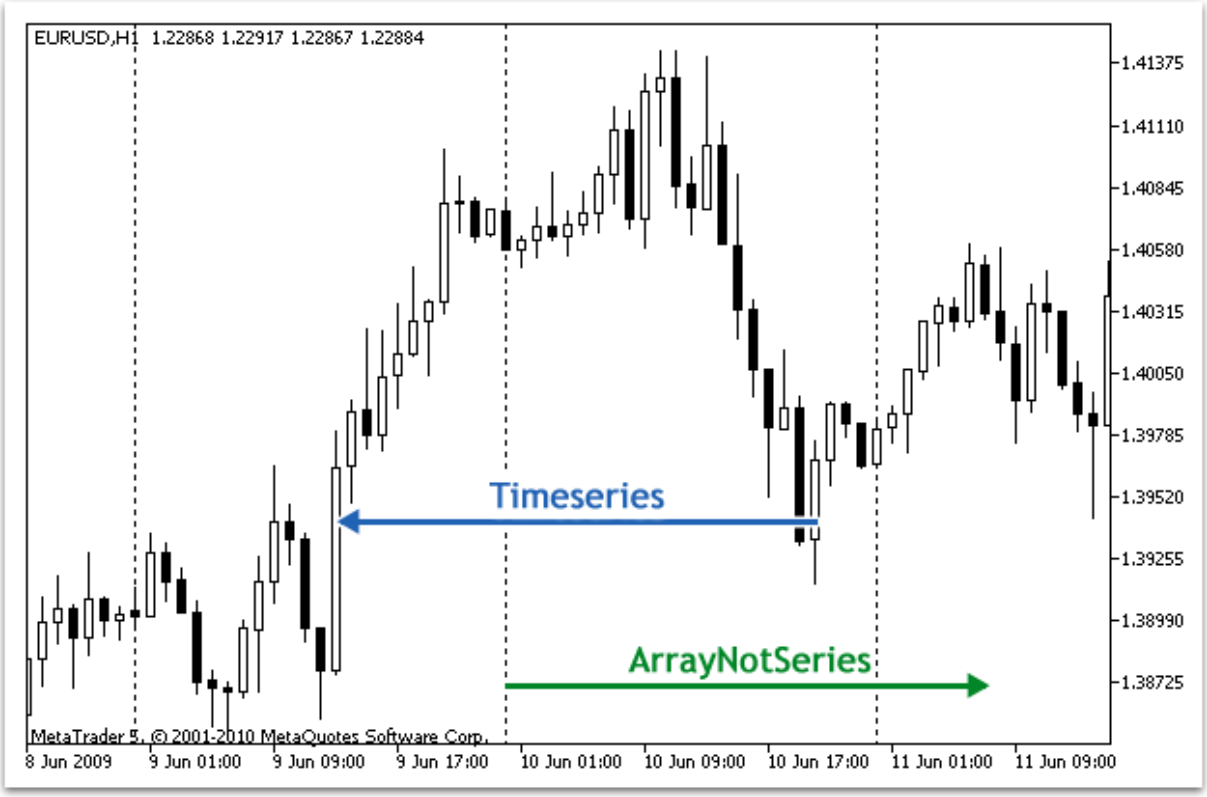
Il existe une **exception importante** à cette règle : si les timeseries et les valeurs de l'indicateur doit être copiées souvent, par exemple à chaque appel à OnTick() dans les Expert Advisors ou à chaque appel de OnCalculate() dans les indicateurs, il est préférable dans ce cas d'utiliser des tableaux distribués de façon statique, car les **opérations d'allocation mémoire** pour les tableaux dynamiques **nécessite davantage de temps**, et ceci aura un effet pendant le test et l'optimisation.

La direction de l'indexation doit être prise en compte lors de l'utilisation de fonctions accédant aux timeseries et aux valeurs de l'indicateur. Tout est décrit dans la section Direction d'Indexation dans les Tableaux, les Buffers et les Timeseries.

L'accès aux données d'un indicateur et aux timeseries est implémenté que les données demandées soient prêtes ou pas (c'est le fameux accès asynchrone). Ceci est d'une grande importance pour le calcul d'un indicateur personnalisé, donc, s'il n'y a pas de données, les fonctions de type *Copy...()* retournent immédiatement une erreur. Cependant, lors de l'accès depuis des Expert Advisors et des scripts, plusieurs tentatives pour obtenir les données sont effectuées dans un court laps de temps, prévu pour donner le temps nécessaire pour télécharger les timeseries demandées ou pour calculer les valeurs de l'indicateur.

La section Organiser l'Accès aux Données décrit le détails pour récupérer, stocker et demander les données dans le terminal client MetaTrader 5.

© 2000-2024, MetaQuotes Ltd.



Il est historiquement accepté que l'accès aux données de prix dans un tableau est effectué depuis la fin des données. Physiquement, les nouvelles données sont toujours écrites à la fin du tableau, mais l'indice du tableau est toujours égal à 0. L'indice 0 dans une timeserie correspond aux données de la barre courante, c'est à dire la barre qui correspond à l'intervalle de temps non encore fini pour la période en cours.

Une période est la période de temps pendant laquelle une seule barre de prix est formée. Il y a 21 périodes standards prédéfinies.

Fonction	Action
SeriesInfoInteger	Retourne l'état des données historiques
Bars	Retourne le nombre de barres dans l'historique pour le symbole et la période spécifiés
BarsCalculated	Retourne le nombre de données calculées dans un buffer d'indicateur ou -1 en cas d'erreur (les données n'ont pas encore été calculées.
IndicatorCreate	Retourne le handle sur l'indicateur technique spécifié créé avec un tableau de paramètres de type MqlParam
IndicatorParameters	A partir du handle spécifié, retourne le nombre de paramètres d'entrée de l'indicateur, ainsi que les valeurs et les types des paramètres
IndicatorRelease	Supprime le handle d'un indicateur et libère son bloc de calcul s'il n'est plus utilisé
CopyBuffer	Retourne dans un tableau les données d'un buffer spécifié de l'indicateur spécifié

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
CopyRates	Retourne dans un tableau les données historiques de la structure Rates pour un symbole et une période spécifiés
CopySeries	Retourne les timseries chronologiques de la structure MqlRates pour le symbole et la période spécifiés et pour la quantité demandée
CopyTime	Retourne dans un tableau les données historiques des heures d'ouverture des barres pour un symbole et une période spécifiés
CopyOpen	Retourne dans un tableau les données historiques des prix d'ouverture des barres pour un symbole et une période spécifiés
CopyHigh	Retourne dans un tableau les données historiques des prix les plus hauts des barres pour un symbole et une période spécifiés
CopyLow	Retourne dans un tableau les données historiques des prix les plus bas des barres pour un symbole et une période spécifiés
CopyClose	Retourne dans un tableau les données historiques des prix de clôture des barres pour un symbole et une période spécifiés
CopyTickVolume	Retourne dans un tableau les données historiques des volumes des ticks pour un symbole et une période spécifiés
CopyRealVolume	Retourne dans un tableau les données historiques des volumes des trades pour un symbole et une période spécifiés
CopySpread	Retourne dans un tableau les données historiques des spreads pour un symbole et une période spécifiés
CopyTicks	Copie les ticks au format MqlTick dans le tableau ticks_array
CopyTicksRange	Copie les ticks au format MqlTick de l'intervalle de dates spécifié dans le tableau ticks_array
iBars	Retourne le nombre de barres dans l'historique pour le symbole et la période spécifiés
iBarShift	Retourne l'indice de la barre correspondant à l'heure spécifiée
iClose	Retourne le prix de clôture (Close) de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant
iHigh	Retourne le prix le plus haut (High) de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant
iHighest	Retourne l'indice de la valeur la plus haute trouvée sur le graphique correspondant (décalage relatif à la barre courante)
iLow	Retourne le prix le plus bas (Low) de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant
iLowest	Retourne l'indice de la valeur la plus basse trouvée sur le graphique correspondant (décalage relatif à la barre courante)

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
iOpen	Retourne le prix d'ouverture (Open) de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant
iTime	Retourne l'heure d'ouverture de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant
iTickVolume	Retourne le volume de ticks de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant
iRealVolume	Retourne le volume réel de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant
iVolume	Retourne le volume de ticks de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant
iSpread	Retourne la valeur du spread de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant

En dépit du fait qu'en utilisant la fonction ArraySetAsSeries() il est possible de configurer l'accès aux éléments d'un tableau de la même façon que dans une timeserie, il faut se souvenir que les éléments d'un tableau sont stockés physiquement dans un seul ordre - seule la direction d'indexation change. Pour le démontrer, écrivons un exemple :

```
   datetime TimeAsSeries[];
//--- définit l'accès au tableau comme dans une série chronologique (timeserie)
   ArraySetAsSeries(TimeAsSeries,true);
   ResetLastError();
   int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
   if(copied<=0)
     {
      Print("L'opération de copie des heures d'ouverture des 10 dernières barres a éch
      return;
     }
   Print("TimeCurrent =",TimeCurrent());
   Print("ArraySize(Time) =",ArraySize(TimeAsSeries));
   int size=ArraySize(TimeAsSeries);
   for(int i=0;i<size;i++)
     {
      Print("TimeAsSeries["+i+"] =",TimeAsSeries[i]);
     }

   datetime ArrayNotSeries[];
   ArraySetAsSeries(ArrayNotSeries,false);
   ResetLastError();
   copied=CopyTime(NULL,0,0,10,ArrayNotSeries);
   if(copied<=0)
     {
      Print("L'opération de copie des heures d'ouverture des 10 dernières barres a éch
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   size=ArraySize(ArrayNotSeries);
   for(int i=size-1;i>=0;i--)
     {
      Print("ArrayNotSeries["+i+"] =",ArrayNotSeries[i]);
     }
```

Le résultat obtenu ressemblera à ceci :

```
TimeCurrent = 2009.06.11 14:16:23
ArraySize(Time) = 10
TimeAsSeries[0] = 2009.06.11 14:00:00
TimeAsSeries[1] = 2009.06.11 13:00:00
TimeAsSeries[2] = 2009.06.11 12:00:00
TimeAsSeries[3] = 2009.06.11 11:00:00
TimeAsSeries[4] = 2009.06.11 10:00:00
TimeAsSeries[5] = 2009.06.11 09:00:00
TimeAsSeries[6] = 2009.06.11 08:00:00
TimeAsSeries[7] = 2009.06.11 07:00:00
TimeAsSeries[8] = 2009.06.11 06:00:00
TimeAsSeries[9] = 2009.06.11 05:00:00

ArrayNotSeries[9] = 2009.06.11 14:00:00
ArrayNotSeries[8] = 2009.06.11 13:00:00
ArrayNotSeries[7] = 2009.06.11 12:00:00
ArrayNotSeries[6] = 2009.06.11 11:00:00
ArrayNotSeries[5] = 2009.06.11 10:00:00
ArrayNotSeries[4] = 2009.06.11 09:00:00
ArrayNotSeries[3] = 2009.06.11 08:00:00
ArrayNotSeries[2] = 2009.06.11 07:00:00
ArrayNotSeries[1] = 2009.06.11 06:00:00
ArrayNotSeries[0] = 2009.06.11 05:00:00
```

Comme nous le voyons dans la sortie ci-dessus, au fur et à mesure que l'indice du tableau TimeAsSeries augmente, la valeur du temps de l'indice baisse, c'est à dire que nous nous déplaçons du présent vers le passé. Pour le tableau classique ArrayNotSeries, le résultat est différent - au fur et à mesure que l'indice augmente, nous nous déplacçons du passé vers le présent.

Voir Aussi

ArrayIsDynamic, ArrayGetAsSeries, ArraySetAsSeries, ArrayIsSeries

© 2000-2024, MetaQuotes Ltd.

Direction d'Indexation dans les Tableaux, les Buffers et les Timeseries

L'indexation par défaut de tous les tableaux et buffers d'un indicateur se fait de gauche à droite. L'indice du premier élément est toujours égal à 0. Le tout premier élément d'un tableau ou du buffer d'un indicateur avec l'indice 0 est donc par défaut à l'extrême gauche, tandis que le tout dernier élément est à l'extrême droite.

Le buffer d'un indicateur est un tableau dynamique d'éléments de double, dont la taille est gérée par le terminal client, afin qu'il corresponde toujours au nombre de barres sur lesquelles l'indicateur est calculé. Un tableau dynamique classique d'éléments de type double est assigné comme buffer de l'indicateur avec la fonction SetIndexBuffer(). Les buffers d'un indicateur ne nécessitent pas de définir leurs tailles avec la fonction ArrayResize() - ceci sera fait par le système d'exécution du terminal.

Les timeseries sont des tableaux avec une indexation inversée, c'est à dire que le premier élément d'une timeserie est à l'extrême droite, et le dernier élément est à l'extrême gauche. Les timeseries étant utilisées pour stocker les historiques des prix et pour contenir les informatiosns sur les heures, nous pouvons dire que les données les plus récentes sont placées à l'extrême droite des timeseries, tandis que les données les plus anciennes sont à l'extrême gauche.

L'élément avec l'indice 0 d'une timeserie contient donc les informations de la dernière cotation d'un symbole. Si une timeserie contient des données d'une période D1, les données encore incomplètes du jour courant sont situées à la position 0, et la position avec l'indice 1 contient les données de la veille.

Changer la Direction d'Indexation

La fonction ArraySetAsSeries() permet de changer la méthode d'accès aux éléments d'un tableau dynamique ; l'ordre physique de stockage des données dans la mémoire de l'ordinateur n'est pas changé. La fonction change simplement la méthode d'adressage des éléments du tableau, afin que lors de la copie d'un tableau dans un autre avec la fonction ArrayCopy(), le contenu du tableau destination ne dépendra pas de la direction d'indexation du tableau source.

La direction d'indexation ne peut pas être changée pour les tableaux distribués de façon statique. Même si un tableau a été passé en paramètre d'une fonction, la tentative de changer la direction d'indexation dans cette fonction n'aura aucun effet.

Pour les buffers d'un indicateur, comme pour les tableaux classiques, la direction d'indexation peut également être définie en arrière (comme pour les timeseries), c'est à dire que la référence à la position 0 dans le buffer de l'indicateur signifiera la dernière valeur du buffer correspondant de l'indicateur et correspondra à la valeur de l'indicateur sur la dernière barre. La localisation physique des barres de l'indicateur restera tout de même inchangée.

Récupérer les Prix dans les Indicateurs

Chaque indicateur personnalisé doit obligatoirement contenir la fonction OnCalculate(), à laquelle les prix nécessaires pour le calcul des valeurs dans les buffers de l'indicateur sont passés. Il est possible de connaître la direction d'indexation de ces tableaux en utilisant la fonction ArrayGetAsSeries().

Les tableaux passés à la fonction reflètent les prix, c'est à dire que ces tableaux ont le flag des timeseries et que la fonction ArrayIsSeries() retournera true lors de la vérification de ces tableaux.

© 2000-2024, MetaQuotes Ltd.

Cependant, il ne faut jamais vérifier la direction d'indexation uniquement avec la fonction ArrayGetAsSeries().

Pour ne pas être dépendant des valeurs par défaut, ArraySetAsSeries() devrait être appelée inconditionnellement pour les tableaux que vous allez utiliser, et pour définir la direction désirée.

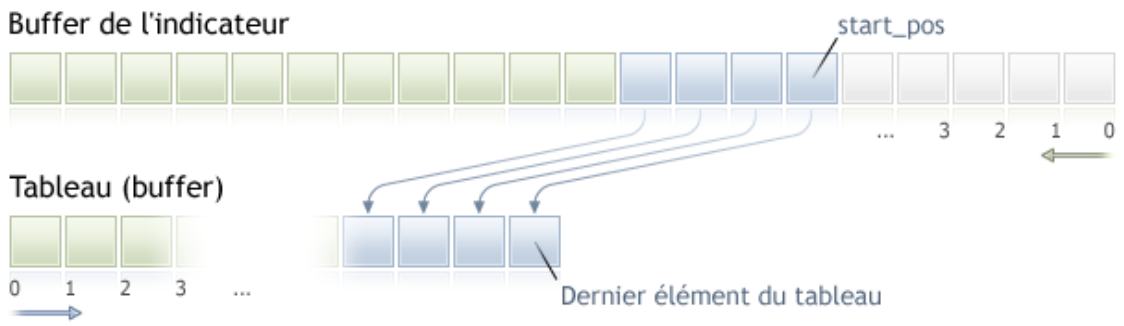
Récupérer les Prix et les Valeurs de l'Indicateur

La direction d'indexation par défaut de tous les tableaux dans les Expert Advisors, dans les indicateurs et dans les scripts est de gauche à droite. Si nécessaire, vous pouvez demander les valeurs des timeseries dans un programme mql5 sur n'importe quel symbole et n'importe quelle période, ainsi que les valeurs des indicateurs calculées sur n'importe quel symbole et n'importe quelle période.

Utilisez les fonctions Copy...() pour cela :

- CopyBuffer - copie les valeurs du buffer de l'indicateur dans un tableau d'éléments de type double ;
- CopyRates - copies l'historique des prix dans un tableau de structures MqlRates ;
- CopyTime - copie les valeurs de temps dans un tableau d'éléments de type datetime ;
- CopyOpen - copie les valeurs d'ouverture (Open) dans un tableau d'éléments de type double ;
- CopyHigh - copie les valeurs des prix les plus hauts (High) dans un tableau d'éléments de type double ;
- CopyLow - copie les valeurs des prix les plus bas (Low) dans un tableau d'éléments de type double ;
- CopyClose - copie les valeurs des prix de clôture (Close) dans un tableau d'éléments de type double ;
- CopyTickVolume - copie les volumes des ticks dans un tableau d'éléments de type long ;
- CopyRealVolume - copie les volumes réels dans un tableau d'éléments de type long ;
- CopySpread - copie l'historique des spreads dans un tableau d'éléments de type int ;

Toutes ces fonctions fonctionnent de la même façon. Considérons le mécanisme d'obtention des données dans l'exemple de CopyBuffer(). Il est implicite que la direction d'indexation des données demandées est celle des timeseries, et que la position avec l'indice 0 stocke les données de la barre non encore complétée. Pour accéder à ces données, vous devez copier le volume nécessaire de données dans le tableau destination, c'est à dire dans le *buffer* du tableau.



Lors de la copie, nous devons spécifier la position de départ dans le tableau source, à partir de laquelle les données seront copiées dans le tableau destination. En cas de succès, le nombre spécifié d'éléments seront copiés dans le tableau destination depuis le tableau source (depuis le buffer de

© 2000-2024, MetaQuotes Ltd.

l'indicateur dans ce cas). Indépendamment de la valeur d'indexation définie dans le tableau destination, la copie est toujours effectuée comme montrée dans la figure ci-dessus.

S'il est attendu que les prix seront traités dans une boucle avec un grand nombre d'itérations, il est conseillé de vérifier la fin forcée du programme en utilisant la fonction IsStopped() :

```
int copied=CopyBuffer(ma_handle,// handle de l'indicateur
                      0,        // L'indice du buffer de l'indicateur
                      0,        // Position de départ pour la copie
                      number,   // Nombre de valeurs à copier
                      Buffer    // Le tableau qui recevra les valeurs
                      );
if(copied<0) return;
int k=0;
while(k<copied && !IsStopped())
  {
   //--- Récupère la valeur pour l'indice k
   double value=Buffer[k];
   // ...
   // utilise la valeur
   k++;
  }
```

Exemple :

```
input int per=10; // période de l'exposant
int ma_handle;    // handle de l'indicateur
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de tick de l'Expert                                     |
//+------------------------------------------------------------------+
void OnTick()
  {
//---
   double ema[10];
   int copied=CopyBuffer(ma_handle,// handle de l'indicateur
                         0,        // indice du buffer de l'indicateur
                         0,        // position de départ à partir de laquelle la copie
                         10,       // nombre de valeurs à copier
                         ema       // tableau destination
                         );
```

© 2000-2024, MetaQuotes Ltd.

```
   if(copied<0) return;
// .... code supplémentaire
  }
```

Voir aussi

Organiser l'Accès aux Données

© 2000-2024, MetaQuotes Ltd.

Organiser l'Accès aux Données

Dans cette section, les questions liées à l'obtention, au stockage et à la demande des données des prix (timeseries) sont considérées.

Recevoir les Données du Serveur de Trades

Avant que les données des prix soient disponibles dans le terminal MetaTrader 5, ils doivent être obtenus et traités. Pour obtenir les données, la connexion au serveur de trades MetaTrader 5 doit être établie. Les données sont obtenues sous la forme de blocs compressés de barres 1 minute depuis le serveur suivant la demande d'un terminal.

Le mécanisme de référence du serveur pour les données ne dépend pas de la façon dont la demande a été initiée - par un utilisateur naviguant dans un graphique ou de façon programmatique dans le langage MQL5.

Stocker les Données Intermédiaires

Les données obtenues du serveur sont automatiquement décompréssées et sauvegardées dans le format intermédiaire HCC. Les données de chaque symbole sont écrites dans un dossier séparé : *répertoire_du_terminal*\bases*nom_du_serveur*\history*nom_du_symbole*. Par exemple, les données de l'EURUSD reçues du serveur MetaQuotes-Demo seront stockées dans *répertoire_du_terminal*\bases\MetaQuotes-Demo\history\EURUSD\.

Les données sont écrites dans des fichiers avec l'extension .hcc. Chaque fichier stocke les données des barres 1 minute pour un an. Par exemple, le fichier nommé 2009.hcc dans le dossier EURUSD contient les barres 1 minutes de l'EURUSD pour l'année 2009. Ces fichiers sont utilisés pour préparer les données des prix pour toutes les périodes et ne sont pas prévus pour être accéder directement.

Obtenir les Données sur une Période Désirée à partir des Données Intermédiaires

Les fichiers HCC intermédiaires sont utilisés comme source de données pour construire les données des prix pour les périodes demandées dans le format HC. Les données au format HC sont des timeseries qui sont préparées au maximum pour un accès rapide. Elles sont créées à la demande d'un graphique ou d'un programme MQL5. Le volume des données ne doit pas excéder la valeur du paramètre "Nb max de barres dans les graphiques". Les données sont stockées pour une utilisation ultérieure dans des fichiers avec l'extension hcc.

Pour économiser des ressources, les données d'une période sont stockées et sauvegardées en RAM uniquement si nécessaire. Si elles ne sont pas appelées pendant un long moment, elles sont libérées de la RAM et sauvegardées dans un fichier. Pour chaque période, les données sont préparées indépendamment du fait qu'il y ait des données prêtes pour les autres périodes ou pas. Les règles de formation et d'accès aux données sont les mêmes pour toutes les périodes. C'est à dire qu'en dépit du fait que les données unitaires stockées au format HCC sont les données 1 minute (M1), la disponibilité des données HCC ne signifie pas la disponibilité des données de la période M1 puisque HC est dans le même volume.

La réception des nouvelles données du serveur appelle automatiquement la mise à jour des données des prix au format HC de toutes les périodes. Cela mène aussi au recalcul de tous les indicateurs qui les utilisent implicitement comme données d'entrée pour les calculs.

© 2000-2024, MetaQuotes Ltd.

Paramètre "Nb max de barres dans le graphique"

Le paramètre "Nb max de barres dans le graphique" restreint le nombre de barres au format HC disponibles dans les graphiques, dans les indicateurs et dans les programmes mql5. Ceci est valide pour toutes les périodes disponibles et sert en tout premier lieu à économiser les ressources de l'ordinateur.

Lorsqu'une grande valeur est mise pour ce paramètre, il faut se souvenir que si un historique profond des données des prix est disponible pour les petites périodes, la mémoire utilisée pour stocker les timeseries et les buffers de l'indicateur peut atteindre des centaines de mega-octets et atteindre la limite de RAM pour le programme du terminal client (2 Go pour les applications 32 bits de MS Windows).

Un changement du paramètre "Nb max de barres dans les graphiques" n'est pris en compte qu'après le redémarrage du terminal client. Le changement de ce paramètre ne cause ni la vérification automatique du serveur pour des données supplémentaires, ni la formation de barres supplémentaires des timeseries. Les données supplémentaires des prix sont demandées au serveur, et les timeseries sont mises à jour en prenant en compte la nouvelle limite, dans le cas du défilement de la zone du graphique où il n'y a pas de données, ou lorsque les données sont demandées par un programme mql5.

Le volume des données demandées au serveur correspond au nombre demandé de barres de cette période avec le paramètre "Nb max de barres dans les graphiques" pris en compte. La restriction donnée par ce paramètre n'est pas stricte, et dans certains cas, le nombre de barres disponibles pour une période peut être supérieur à la valeur actuelle du paramètre.

Disponibilité des Données

La présence de données au format HCC ou même préparée pour être utilisées au format HC ne signifie pas toujours la disponibilité absolue de ces données pour être affichées dans un graphique ou pour être utilisées dans des programmes MQL5.

Lors de l'accès aux données des prix ou aux valeurs d'un indicateur depuis un programme mql5, il faut se souvenir que leurs disponibilités à un certain moment ou à partir d'un certain moment ne sont pas garanties. Ceci est dû au fait que dans le but d'économiser des ressources, la copie complète des données nécessaires pour un programme mql5 n'est pas stocké dans MetaTrader 5 ; seul un accès direct à la base de données du terminal est donné.

L'historique des prix pour toutes les périodes est construit à partir des données communes au format HCC, and toute mise à jour des données du serveur conduit à la mise à jour des données pour toutes les périodes et au recalcul des indicateurs. A cause de cela, l'accès aux aux données peut être fermé, même si ces données n'étaient pas disponibles auparavant.

Synchronisation des Données du Terminal et des Données du Serveur

Puisqu'un programme mql5 peut appeler des données d'un autre symbole et d'une autre période, il y a la possibilité que les données des timeseries nécessaires ne soient pas encore formées dans le terminal ou que les données nécessaires des prix ne sont pas synchronisées avec le serveur de trades. Dans ce cas, prédire le temps de latence est difficile.

Les algorithmes utilisant des boucles "ne-fait-rien" ne sont pas la meilleure solution. La seule exception dans ce cas sont les scripts, car il n'existe pas d'autre choix d'algorithme car ils ne gèrent pas d'évènements. Pour les indicateurs personnalisés, de tels algorithmes, ainsi que n'importe quelles

© 2000-2024, MetaQuotes Ltd.

autres boucles "ne-fait-rien", sont fortement à éviter, car ils mènent à la fin du calcul de tous les indicateurs et tout autre traitement des données des prix du symbole.

Pour les Expert Advisors et les indicateurs, il est préférable d'utiliser le modèle de gestion d'évènements. Si pendant le traitement de l'évènement OnTick() ou OnCalculate(), la réception des données pour les timeseries requises échoue, vous devriez sortir de la fonction de gestion de l'évènement, en vous basant sur la disponibilité de l'accès à l'appel suivant de la fonction.

Exemple de Script d'Ajout dans l'Historique

Considérons l'exemple d'un script qui exécute une demande d'obtention de l'historique pour le symbole sélectionné auprès du serveur de trades. Le script est prévu pour être exécuté dans le graphique d'un symbole sélectionné ; la période n'importe pas car, comme il a été mentionné auparavant, les données des prix sont reçues du serveur de trades sous la forme de données 1 minutes compréssées à partir desquelles les timeseries prédéfinies sont construites.

Il écrit toutes les actions concernant l'obtention des données dans une fonction CheckLoadHistory(symbol, timeframe, start_date) séparée :

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
  }
```

La fonction CheckLoadHistory() est conçue comme fonction universelle qui peut être appelée depuis n'importe quel programme (Expert Advisor, script ou indicateur). Il nécessite donc 3 paramètres d'entrée : nom du symbole, période et date de début pour indiquer le début de l'historique des prix dont vous avez besoin.

Il insère les vérifications nécessaires dans le code de la fonction avant de demander l'historique manquant. Tout d'abord, nous devons nous assurer que le nom du symbole et la période sont corrects :

```
   if(symbol==NULL || symbol==="") symbol=Symbol();
   if(period==PERIOD_CURRENT)      period=Period();
```

Assurons-nous ensuite que le symbole est disponible dans la fenêtre du MarketWatch, c'est à dire que l'historique pour le symbole sera disponible lors de l'envoi d'une demande au serveur de trades. Si un tel symbole n'existe pas dans le MarketWatch, ajoutez-le avec la fonction SymbolSelect().

```
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
    {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
    }
```

Nous devons maintenant récupérer la date de départ de l'historique disponible pour la paire symbole/période donnée. Peut être que la valeur du paramètre d'entrée startdate, passé à CheckLoadHistory(), est dans l'historique disponible ; la demande au serveur de trades n'est donc pas nécessaire. Pour obtenir la toute première date de la paire symbole-période à ce moment, la fonction SeriesInfoInteger() est utilisée avec le modificateur SERIES_FIRSTDATE.

```
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
```

© 2000-2024, MetaQuotes Ltd.

La prochaine vérification importante est le type du programme à partir duquel la fonction est appelée. Notez qu'il n'est pas souhaitable d'envoyer une demande pour mettre à jour les timeseries depuis l'indicateur dans la même période. Le caractère indésirable de la demande de données sur la même paire symbole-période que celle de l'indicateur est conditionné par le fait que la mise à jour des données de l'historique est effectuée dans le même thread que celui où l'indicateur fonctionne. La possibilité de deadlock est donc élevée. Pour le vérifier, utilisez la fonction MQL5InfoInteger() avec le modificateur MQL5_PROGRAM_TYPE.

```
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sym
      return(-4);
```

Si toutes les vérifications ont été réalisées avec succès, effectuez la dernière tentative sans vous référer au serveur de trades. Trouvons d'abord la date de dpart pour laquelle les données 1 minute au format HCC sont disponibles. Récupérez cette valeur en utilisation la fonction SeriesInfoInteger() avec le modificateur SERIES_TERMINAL_FIRSTDATE et comparez-la à nouveau avec la valeur du paramètre start_date.

```
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- il y a des données chargées pour construire les timeseries
      if(first_date>0)
        {
         //--- force la construction des timeseries
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- vérifie la date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
```

Si après toutes les vérifications, le thread d'exécution est toujours dans le corps de la fonction CheckLoadHistory(), cela signifie qu'il est nécessaire de demander les données manquantes des prix au serveur de trades. Retourne d'abord la valeur du paramètre "Nb max de barres dans le graphique" avec la fonction TerminalInfoInteger() :

```
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

Nous en aurons besoin pour éviter de demander des données supplémentaires. Trouvons ensuite la toute première date de l'historique du symbole sur le serveur de trades (indépendamment de la période) en utilisant la fonction SeriesInfoInteger() avec le modificateur SERIES_SERVER_FIRSTDATE.

```
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
```

Puisque la demande est une opération asynchrone, la fonction est appelée dans la boucle avec un court délai de 5 millisecondes jusqu'à ce que la variable first_server_date reçoive une valeur, ou que la boucle d'exécution soit stoppée par l'utilisateur (IsStopped() retournera *true* dans ce cas). Indiquons une valeur correcte de date de départ, à partir de laquelle nous demandons les données des prix au serveur de trades.

```
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
```

© 2000-2024, MetaQuotes Ltd.

```
      Print("Attention : la première date du serveur ",first_server_date," pour ",
symbol," ne correspond pas à la date des premières séries ",first_date);
```

Si la date de départ *first_server_date* du serveur est inférieure à la date de départ *first_date* du symbole au format HCC, l'entrée correspondante sera écrite dans le journal.

Nous sommes maintenant prêt à effectuer une demande au serveur de trades pour demandes les données manquantes des prix. Effectuons la demande sous la forme d'une boucle et commençons à remplir son corps :

```
   while(!IsStopped())
     {
      //1. attendons la synchronisation entre les timeseries reconstruites et l'histo
      //2. récupérons le nombre courant de barres dans cette timeserie
      //    si la barre est supérieure à Max_bars_in_chart, nous pouvons sortir, le t
      //3. récupérons la date de départ first_date dans les timeseries reconstruites
      //    si first_date est inférieure à start_date, nous pouvons sortir, le travai
      //4. demandons au serveur une nouvelle partie de l'historique - 100 barres à pa
     }
```

Les 3 premiers points sont implémentés par des moyens déjà connus.

```
   while(!IsStopped())
     {
      //--- 1. attends que le process de reconstruction des timeseries soit fini
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- 2. demande combien de barres nous avons
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         //--- barres supérieures à celles qui peuvent être dessinées sur le graphiqu
         if(bars>=max_bars) return(-2);
         //--- 3. retourne la date de départ actuelle dans les timeseries
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            // la date de départ est plus récente que celle demandée, la tâche est te
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //4. Demandons au serveur une nouvelle partie de l'historique - 100 barres à pa
     }
```

Il reste le quatrième et dernier point à faire - demander l'historique. Nous ne pouvons pas nous référer à un serveur directement, mais n'importe quelle fonction Copy inities automatiquement l'envoi de la demande à un serveur si l'historique au format HCC n'est pas suffisant. Puisque l'heure de la toute première date dans la variable *first_date* est le critère simple et naturel pour évaluer le degré d'exécution de la demande, la façon la plus simple est donc d'utiliser la fonction CopyTime().

Lors de l'appel aux fonctions de copie des données à partir des timeseries, il est à noter que le paramètre *start* (numéro de la barre à partir de laquelle les données des prix sont à copier) doit toujours être dans l'historique disponible du terminal. Si vous n'avez que 100 barres, il est inutile d'essayer de copier 300 barres à partir de la barre d'indice 500. Un telle demande sera considérée

© 2000-2024, MetaQuotes Ltd.

comme invalide et ne sera pas traitée, c'est à dire qu'aucun historique supplémentaire ne sera chargé du serveur de trades.

c'est pourquoi nous copierons les barres en groupes de 100 en commençant par la barre avec l'indice *bars* . Cela permettra un chargement lissé de l'historique manquant depuis le serveur de trades. En fait, un peu plus de 100 barres seront chargées, car le serveur renvoie un historique plus grand.

```
   int copied=CopyTime(symbol,period,bars,100,times);
```

Après la copie, nous devons analyser le nombre d'éléments copiés. Si la tentative échoue, alors la valeur de *copied* sera égale à 0 et la valeur du compteur fail_cnt sera incrémentée de 1. Après 100 échecs, l'exécution de la fonction sera stoppée.

```
int fail_cnt=0;
...
   int copied=CopyTime(symbol,period,bars,100,times);
   if(copied>0)
     {
      //--- vérification des données
      if(times[0]<=start_date)  return(0);  // la valeur copiée est plus petite, tout
      if(bars+copied>=max_bars) return(-2); // il y a plus de barres que ce qui peut é
      fail_cnt=0;
     }
   else
     {
      //--- pas plus de 100 échecs à la suite
      fail_cnt++;
      if(fail_cnt>=100) return(-5);
      Sleep(10);
     }
```

Donc, non seulement la gestion correcte de la situation correcte à tout moment de l'exécution est implémentée dans la fonction, mais également le code de fin est retourné. Il peut être géré après l'appel à la fonction CheckLoadHistory() pour obtenir des informations supplémentaires. Par exemple, de cette façon :

```
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Symbole inconnu ",InpLoadedSymbol);
      case -2 : Print("Plus de barres que ce qui peut être dessiné dans le graphique")
      case -3 : Print("Exécution stoppée par l'utilisateur");
      case -4 : Print("L'indicateur ne doit pas charger ses propres données");
      case -5 : Print("Echec du chargement");
      case  0 : Print("Toutes les données ont été chargées");
      case  1 : Print("Les données déjà disponibles dans les timeseries sont suffisant
      case  2 : Print("La timeserie est construite à partir des données disponibles du
      default : Print("Résultat de l'exécution non défini");
     }
```

© 2000-2024, MetaQuotes Ltd.

Le code complet de la fonction est disponible dans l'exemple du script qui montre l'organisation correcte de l'accès à n'importe quelle donnée dans la gestion des résultats de la demande.

Code :

```
//+------------------------------------------------------------------+
//|                                              TestLoadHistory.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
//--- paramètres d'entrée
input string          InpLoadedSymbol="NZDUSD";   // symbole à charger
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1;  // période à charger
input datetime        InpStartDate=D'2006.01.01'; // date de début
//+------------------------------------------------------------------+
//| Fonction de démarrage du script        |
//+------------------------------------------------------------------+
void OnStart()
  {
   Print("Démarre le chargement",InpLoadedSymbol+","+GetPeriodName(InpLoadedPeriod),"
//---
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Symbole inconnu ",InpLoadedSymbol);
      case -2 : Print("Nombre de barres demandées supérieure au nombre max de barres 
      case -3 : Print("Le programme a été stoppé");
      case -4 : Print("L'indicateur ne doit pas charger ses propres données");
      case -5 : Print("Echec du chargement");
      case  0 : Print("Chargement réussi");
      case  1 : Print("Chargé précédemment");
      case  2 : Print("Chargé précédemment et construit");
      default : Print("Résultat inconnu");
     }
//---
   datetime first_date;
   SeriesInfoInteger(InpLoadedSymbol,InpLoadedPeriod,SERIES_FIRSTDATE,first_date);
   int bars=Bars(InpLoadedSymbol,InpLoadedPeriod);
   Print("Première date ",first_date," - ",bars," barres");
//---
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
```

```
  {
   datetime first_date=0;
   datetime times[100];
//--- vérifie le symbole et la période
   if(symbol==NULL || symbol==""") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
//--- vérifie si le symbole est sélectionné dans le Market Watch
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
//--- vérifie si les données sont présentes
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
//--- ne demande pas le chargement de ses propres données si c'est un indicateur
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sym
      return(-4);
//--- deuxième tentative
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- il y a des données chargées pour construire les timeseries
      if(first_date>0)
        {
         //--- force la construction des timeseries
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- vérifie la date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
//--- nb max de barres dans le graphique depuis les options du terminal
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- charge les informations de l'historique du symbole
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
//--- fixe la date de début du chargement
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Attention : la première date du serveur ",first_server_date," pour ",symb
            " ne correspond pas ) la date des premières séries",first_date);
//--- chargement des données pas à pas
   int fail_cnt=0;
   while(!IsStopped())
     {
      //--- attend la construction des timeseries
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- demande la construction des barres
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         if(bars>=max_bars) return(-2);
         //--- demande la première date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //--- la copie de la partie suivante force le chargement des données
      int copied=CopyTime(symbol,period,bars,100,times);
      if(copied>0)
        {
         //--- vérifie les données
         if(times[0]<=start_date)  return(0);
         if(bars+copied>=max_bars) return(-2);
         fail_cnt=0;
        }
      else
        {
         //--- pas plus de 100 échecs
         fail_cnt++;
         if(fail_cnt>=100) return(-5);
         Sleep(10);
        }
     }
//--- stoppé
   return(-3);
  }
//+------------------------------------------------------------------+
//| Retourne une chaîne de caractères correspondant à la période |
//+------------------------------------------------------------------+
string GetPeriodName(ENUM_TIMEFRAMES period)
  {
   if(period==PERIOD_CURRENT) period=Period();
//---
   switch(period)
     {
      case PERIOD_M1:  return("M1");
      case PERIOD_M2:  return("M2");
      case PERIOD_M3:  return("M3");
      case PERIOD_M4:  return("M4");
      case PERIOD_M5:  return("M5");
      case PERIOD_M6:  return("M6");
      case PERIOD_M10: return("M10");
      case PERIOD_M12: return("M12");
      case PERIOD_M15: return("M15");
      case PERIOD_M20: return("M20");
      case PERIOD_M30: return("M30");
```

© 2000-2024, MetaQuotes Ltd.

```
      case PERIOD_H1:  return("H1");
      case PERIOD_H2:  return("H2");
      case PERIOD_H3:  return("H3");
      case PERIOD_H4:  return("H4");
      case PERIOD_H6:  return("H6");
      case PERIOD_H8:  return("H8");
      case PERIOD_H12: return("H12");
      case PERIOD_D1:  return("Daily");
      case PERIOD_W1:  return("Weekly");
      case PERIOD_MN1: return("Monthly");
     }
//---
   return("période inconnue");
  }
```

© 2000-2024, MetaQuotes Ltd.

SeriesInfoInteger

Retourne l'information sur l'état des données historiques. Il existe 2 variantes d'appel à la fonction.

Retourne directement la valeur de la propriété.

```
long  SeriesInfoInteger(
   string                      symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES             timeframe,        // période
   ENUM_SERIES_INFO_INTEGER    prop_id,          // identifiant de la propriété
   );
```

Retourne true ou false suivant le succès de l'exécution de la fonction.

```
bool  SeriesInfoInteger(
   string                      symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES             timeframe,        // période
   ENUM_SERIES_INFO_INTEGER    prop_id,          // identifiant de la propriété
   long&                       long_var          // variable pour obtenir les informatio
   );
```

Parameters

symbol_name

[in] Nom du symbole.

timeframe

[in] Période.

prop_id

[in] Identifiant de la propriété demandée, valeur de l'énumération ENUM_SERIES_INFO_INTEGER.

long_var

[out] Variable dans laquelle la valeur de la propriété demandée est placée.

Valeur de Retour

Dans le premier cas, retourne une valeur de type long.

Pour le deuxième cas, retourne true si la propriété spécifiée est disponible et que sa valeur a été placée dans la variable long_var, sinon retourne false. Pour plus de détails sur l'erreurr, appelez la fonction GetLastError().

Exemple :

```
void OnStart()
  {
//---
   Print("Le nombre total de barres pour la paire symbole-période à ce moment = ",
         SeriesInfoInteger(Symbol(),Period(),SERIES_BARS_COUNT));

   Print("La première date pour la paire symbole-période à ce moment = ",
         (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_FIRSTDATE));
```

© 2000-2024, MetaQuotes Ltd.

```
  Print("La première date dans l'historique pour la paire symbole-période sur le serv
        (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_SERVER_FIRSTDATE));

  Print("Les données du symbole sont synchronisées = ",
        (bool)SeriesInfoInteger(Symbol(),Period(),SERIES_SYNCHRONIZED));
}
```

© 2000-2024, MetaQuotes Ltd.

Bars

Retourne le nombre de barres dans l'historique pour le symbole et la période spécifiés. Il existe 2 variantes d'appel à la fonction.

Demande toutes les barres de l'historique

```
int  Bars(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  timeframe         // période
   );
```

Demande les barres de l'historique pour l'intervalle de temps donné

```
int  Bars(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  timeframe,        // période
   datetime         start_time,       // date et heure de début
   datetime         stop_time         // date et heure de fin
   );
```

Parameters

symbol_name

[in] Nom du symbole.

timeframe

[in] Période.

start_time

[in] Heure de la barre correspondant au premier élément.

stop_time

[in] Heure de la barre correspondant au dernier élément.

Valeur de Retour

Si les paramètres start_time et stop_time sont définis, la fonction retourne le nombre de barres dans l'intervalle spécifié, sinon elle retourne le nombre total de barres.

Note

Si les données des timeseries avec les paramètres spécifiés ne sont pas formés dans le terminal au moment de l'appel à la fonction Bars(), ou que les données des timeseries ne sont pas synchronisées avec le serveur de trades au moment de l'appel à la fonction, la fonction retourne 0.

Lors de la demande du nombre de barre dans un intervalle de temps spécifié, seules sont considérées les barres dont l'heure d'ouverture est dans l'intervalle. Par exemple, si le jour courant de la semaine est un samedi et que la requête est faite pour le nombre de barres W1 avec start_time=Last_Tuesday et stop_time=Last_Friday, la fonction retournera 0, puisque l'heure d'ouverture de la période W1 est toujours le dimanche et qu'aucune barre W1 ne tombe dans l'intervalle spécifié.

Exemple d'appel pour obtenir le nombre total de barres dans l'historique :

© 2000-2024, MetaQuotes Ltd.

```
int bars=Bars(_Symbol,_Period);
if(bars>0)
  {
   Print("Nombre de barres dans l'historique du terminal pour le symbole et la pér
  }
else  // aucune barre n'est disponible
  {
   //--- les données du symbole ne sont peut être pas synchronisées avec les donnée
   bool synchronized=false;
   //--- compteur de boucle
   int attempts=0;
   // effectue 5 tentatives pour attendre la synchronisation
   while(attempts<5)
     {
      if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCHRONIZED))
        {
         //--- synchronisation terminée, sortie
         synchronized=true;
         break;
        }
      //--- augmente le compteur
      attempts++;
      //--- attends 10 millisecondes jusqu'à la prochaine itération
      Sleep(10);
     }
   //--- sort de la boucle après la synchronisation
   if(synchronized)
     {
      Print("Nombre de barres dans l'historique du terminal pour le symbole et la p
      Print("La première date dans l'historique du terminal pour le symbole et la p
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
      Print("La première date dans l'historique pour le symbole sur le serveur = ",
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
     }
   //--- la synchronisation des données ne s'est pas produite
   else
     {
      Print("Echec de l'obtention du nombre de barres pour ",_Symbol);
     }
  }
```

Exemple d'appel pour obtenir le nombre de barres dans l'intervalle spécifié :

```
int n;
datetime date1 = D'2016.09.02 23:55'; // Vendredi
datetime date2 = D'2016.09.05 00:00'; // Lundi
datetime date3 = D'2016.09.08 00:00'; // Jeudi
//---
n=Bars(_Symbol,PERIOD_H1,D'2016.09.02 02:05',D'2016.09.02 10:55');
```

© 2000-2024, MetaQuotes Ltd.

```
Print("Nombre de barres : ",n); // Sortie : "Nombre de barres : 8", la barre H2 est
n=Bars(_Symbol,PERIOD_D1,date1,date2);
Print("Nombre de barres : ",n); // Sortie : "Nombre de barres : 1", puisque l'heure
n=Bars(_Symbol,PERIOD_W1,date2,date3);
Print("Nombre de barres : ",n); // Sortie : "Nombre de barres : 0", puisqu'aucune b
```

Voir aussi

Fonctions de Gestion d'Evènement

© 2000-2024, MetaQuotes Ltd.

BarsCalculated

Retourne le nombre de données calculées pour l'indicateur spécifié.

```
int  BarsCalculated(
   int       indicator_handle,     // handle de l'indicateur
   );
```

Parameters

indicator_handle

[in] Le handle sur l'indicateur, retourné par la fonction correspondante de l'indicateur.

Valeur de Retour

Retourne le nombre de données calculées dans le buffer de l'indicateur ou -1 en cas d'erreur (les données ne sont pas encore calculées)

Note

La fonction est utile lorsqu'il est nécessaire de récupérer les données de l'indicateur immédiatement après sa création (le handle de l'indicateur est disponible).

Exemple :

```
void OnStart()
  {
   double Ups[];
//--- définit l'ordre du tableau en mode timeserie
   ArraySetAsSeries(Ups,true);
//--- crée le handle pour l'indicateur Fractal
   int FractalsHandle=iFractals(NULL,0);
//--- réinitialise le code d'erreur
   ResetLastError();
//--- essaye de copier les valeurs de l'indicateur
   int i,copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Sleep(50);
      for(i=0;i<100;i++)
        {
         if(BarsCalculated(FractalsHandle)>0)
            break;
         Sleep(50);
        }
      copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
      if(copied<=0)
        {
         Print("Echec de la copie des fractales supérieures. Erreur = ",GetLastError()
         "i = ",i,"    copiés = ",copied);
         return;
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      else
        Print("Fractales supérieures copiées",
        "i = ",i,"    copiés = ",copied);
    }
  else Print("Fractales supérieures copiées. ArraySize = ",ArraySize(Ups));
}
```

© 2000-2024, MetaQuotes Ltd.

IndicatorCreate

La fonction retourne le handle de l'indicateur technique spécifié et créé à partir du tableau de paramètres de type MqlParam.

```
int  IndicatorCreate(
   string           symbol,                          // nom du symbole
   ENUM_TIMEFRAMES  period,                          // période
   ENUM_INDICATOR   indicator_type,                  // type d'indicateur de l'énumé
   int              parameters_cnt=0,                // nombre de paramètres
   const MqlParam&  parameters_array[]=NULL,         // tableau de paramètres
   );
```

Parameters

symbol

[in] Nom du symbole sur les données duquel l'indicateur sera calculé. NULL signifie le symbole courant.

period

[in] La valeur de la période peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifie la période courante.

indicator_type

[in] Type de l'indicateur, peut être l'une des valeurs de l'énumération ENUM_INDICATOR.

parameters_cnt

[in] Le nombre de paramètres passés dans le tableau parameters_array[]. Les éléments du tableau sont du type spécial MqlParam. Par défaut, aucun paramètre n'est passé. Si vous spécifiez un nombre positif de paramètres, le paramètre *parameters_array* est obligatoire. Vous ne pouvez pas passer plus de 64 paramètres.

parameters_array[]=NULL

[in] Un tableau d'éléments de type MqlParam, dont les éléments contiennent le type et la valeur de chaque paramètre d'entrée d'un indicateur technique.

Valeur de Retour

Retourne le handle de l'indicateur technique spécifié, retourne INVALID_HANDLE en cas d'échec.

Note

Si le handle de l'indicateur de type IND_CUSTOM est créé, le champ *type* du premier élément du tableau des paramètres d'entrée *parameters_array* doit avoir la valeur TYPE_STRING de l'énumération ENUM_DATATYPE, et le champ *string_value* du premier élément doit contenir le nom de l'indicateur personnalisé. L'indicateur personnalisé doit être compilé (fichier avec une extension EX5) et doit être situé dans le répertoire MQL5/Indicators du terminal client ou dans un sous-répertoire.

Les indicateurs nécessitant un test sont définis automatiquement à partir de l'appel à la fonction iCustom(), si le paramètre correspondant est défini avec constant string. Pour tous les autres cas (utilisation de la fonction IndicatorCreate() ou utilisation d'une chaîne de caractères (string) non

© 2000-2024, MetaQuotes Ltd.

constante dans le paramètre définissant le nom de l'indicateur), la propriété #property tester_indicator est nécessaire :

```
#property tester_indicator "indicator_name.ex5"
```

Si la première forme de l'appel est utilisée dans un indicateur personnalisé, vous pouvez indiquer en plus comme dernier paramètre sur quelles données l'indicateur sera calculé. Si le paramètre "Appliquer à" n'est pas spécifié explicitement, le calcul sera basé par défaut sur les valeurs PRICE_CLOSE.

Exemple :

```
void OnStart()
  {
   MqlParam params[];
   int      h_MA,h_MACD;
//--- crée iMA("EURUSD",PERIOD_M15,8,0,MODE_EMA,PRICE_CLOSE);
   ArrayResize(params,4);
//--- configure ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=8;
//--- configure ma_shift
   params[1].type         =TYPE_INT;
   params[1].integer_value=0;
//--- configure ma_method
   params[2].type         =TYPE_INT;
   params[2].integer_value=MODE_EMA;
//--- configure applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=PRICE_CLOSE;
//--- crée la MA
   h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);
//--- crée iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA);
   ArrayResize(params,4);
//--- configure la ma_period rapide
   params[0].type         =TYPE_INT;
   params[0].integer_value=12;
//--- configure la ma_period lente
   params[1].type         =TYPE_INT;
   params[1].integer_value=26;
//--- configure la période de lissage
   params[2].type         =TYPE_INT;
   params[2].integer_value=9;
//--- configure le handle de l'indicateur comme applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=h_MA;
//--- crée le MACD à partir de la moyenne mobile
   h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- utilisation des indicateurs
//--- . . .
```

© 2000-2024, MetaQuotes Ltd.

```
//--- libère les indicateurs (en premier h_MACD)
   IndicatorRelease(h_MACD);
   IndicatorRelease(h_MA);
  }
```

© 2000-2024, MetaQuotes Ltd.

IndicatorParameters

A partir du handle spécifié, retourne le nombre de paramètres d'entrée de l'indicateur, ainsi que les valeurs et les types des paramètres.

```
int  IndicatorParameters(
   int               indicator_handle,      // handle de l'indicateur
   ENUM_INDICATOR&   indicator_type,        // une variable pour recevoir le type de l'
   MqlParam&         parameters[]           // un tableau pour recevoir les paramètres
   );
```

Parameters

indicator_handle

[in] Le handle de l'indicateur, pour lequel vous devez connaître le nombre de paramètres sur lequel il est calculé.

indicator_type

[out] Une variable du type ENUM_INDICATOR, dans laquelle le type de l'indicateur sera écrit.

parameters[]

[out] Un tableau dynamique pour recevoir les valeurs de type MqlParam, dans lequel la liste des paramètres de l'indicateur seront écrits. La taille du tableau est retournée par la fonction IndicatorParameters().

Valeur de Retour

Le nombre de paramètres d'entrées de l'indicateur avec le handle spécifié. En cas d'erreur, retourne -1. Pour plus de détails sur l'erreur, appelez la fonction GetLastError().

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {

//--- Le nombre de fenêtres sur le graphique (au moins une fenêtre principale est tou
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
//--- Parcours les fenêtres du graphique
   for(int w=0;w<windows;w++)
     {
      //--- Le nombre d'indicateurs dans cette fenêtre/sous-fenêtre
      int total=ChartIndicatorsTotal(0,w);
      //--- Prends tous les indicateurs dans la fenêtre
      for(int i=0;i<total;i++)
        {
         //--- Retourne le nom court de l'indicateur
         string name=ChartIndicatorName(0,w,i);
         //--- Retourne le handle de l'indicateur
         int handle=ChartIndicatorGet(0,w,name);
```

© 2000-2024, MetaQuotes Ltd.

```
        //--- Ajoute au journal
        PrintFormat("Fenêtre=%d,  indicateur #%d,  handle=%d",w,i,handle);
        //---
        MqlParam parameters[];
        ENUM_INDICATOR indicator_type;
        int params=IndicatorParameters(handle,indicator_type,parameters);
        //--- L'en-tête du message
        string par_info="Nom court "+name+", type "
                        +EnumToString(ENUM_INDICATOR(indicator_type))+"\r\n";
        //---
        for(int p=0;p<params;p++)
          {
           par_info+=StringFormat("paramètre %d: type=%s, long_value=%d, double_value
                                  p,
                                  EnumToString((ENUM_DATATYPE)parameters[p].type),
                                  parameters[p].integer_value,
                                  parameters[p].double_value,
                                  parameters[p].string_value
                                  );
          }
        Print(par_info);
       }
     //--- Fini pour tous les indicateurs dans la fenêtre
    }
//---
  }
```

Voir aussi

ChartIndicatorGet()

© 2000-2024, MetaQuotes Ltd.

IndicatorRelease

La fonction supprime le handle d'un indicateur et libère son bloc de calcul s'il n'est plus utilisé.

```
bool  IndicatorRelease(
   int       indicator_handle     // handle de l'indicateur
   );
```

Valeur de Retour

Retourne true en cas de succès, false sinon.

Note

La fonction permet de supprimer le handle d'in indicateur s'il n'est plus utilisé afin de libérer de la mémoire. Le handle est supprimé immédiatement, le bloc de calcul est supprimé ensuite (s'il n'est plus utilisé).

Lors de l'utilisation du strategy tester, la fonction IndicatorRelease() n'est pas exécutée.

Exemple :

```
//+------------------------------------------------------------------+
//|                                        Test_IndicatorRelease.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- paramètres d'entrée
input int                MA_Period=15;
input int                MA_shift=0;
input ENUM_MA_METHOD     MA_smooth=MODE_SMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
//--- reçevra le handle de l'indicateur
int MA_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le handle de l'indicateur
   MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,PRICE_CLOSE);
//--- supprime la variable globale
   if(GlobalVariableCheck("MA_value"))
      GlobalVariableDel("MA_value");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de tick de l'Expert                                     |
```

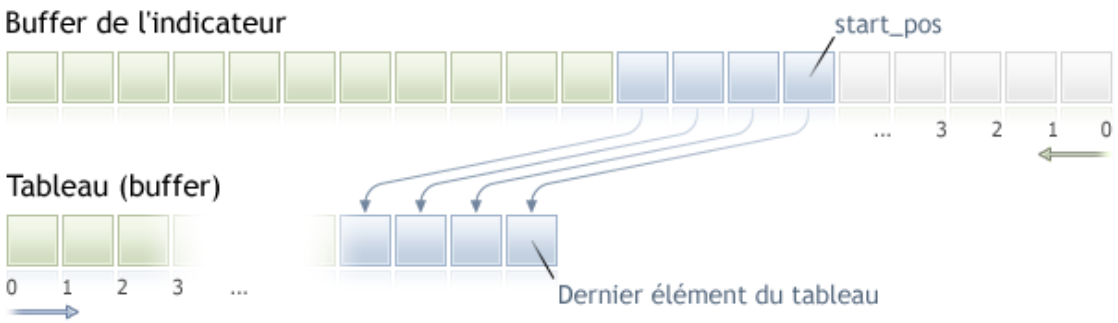
© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnTick()
  {
//--- si la variable globale n'existe pas
   if(!GlobalVariableCheck("MA_value"))
     {
      //--- obtient la valeur de l'indicateur sur les deux dernières barres
      if(MA_handle!=INVALID_HANDLE)
        {
         //--- tableau dynamique pour les valeurs de l'indicateur
         double values[];
         if(CopyBuffer(MA_handle,0,0,2,values)==2 && values[0]!=EMPTY_VALUE)
           {
            //--- stocke dans la variable globale la valeur de la dernière barre
            if(GlobalVariableSet("MA_value",values[0]))
              {
               //--- libère le handle de l'indicateur
               if(!IndicatorRelease(MA_handle))
                  Print("Echec de IndicatorRelease(). Erreur ",GetLastError());
               else MA_handle=INVALID_HANDLE;
              }
            else
               Print("Echec de GlobalVariableSet. Erreur ",GetLastError());
           }
        }
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

CopyBuffer

Retourne la quantité nécessaire de données du buffer spécifié d'un certain indicateur.



Le compte des éléments de données copiées (buffer de l'indicateur avec l'indice buffer_num) à partir du point de départ est effectué du présent vers le passé, c'est à dire que la position de départ de 0 signifie la barre courante (valeur de l'indicateur pour la barre courante).

Lors de la copie d'une quantité inconnue de données, il est recommandé d'utiliser un tableau dynamique buffer[] comme buffer destination, car la fonction CopyBuffer() essaye d'allouer la taille du tableau destination à la taille des données copiées. Si le buffer d'un indicateur (tableau pré-alloué pour stocker les valeurs de l'indicateur par la fonction SetIndexBufer()) est utilisé comme le tableau destination buffer[], la copie partielle est autorisée. Un exemple est disponible dans l'indicateur personnalisé Awesome_Oscillator.mql5 dans le package standard du terminal.

Si vous devez effectuer une copie partielle des valeurs de l'indicateur dans un autre tableau (autre que le buffer d'un indicateur), vous devriez utiliser un tableau intermédiaire dans lequel le nombre désiré est copié. Après cela, copiez le nombre désiré d'éléments depuis ce tableau intermédiaire dans le tableau destination.

Si vous connaissez la quantité de données à copier, il est préférable d'utiliser un buffer alloué de façon statique, afin d'éviter une allocation mémoire excessive.

Ceci quelque soit la propriété du tableau destination - as_series=true ou as_series=false. Les données seront copiées afin que le plus vieil élément soit situé au départ de la mémoire physique allouée pour le tableau. Il existe 3 variantes d'appel à la fonction.

Appel avec la première position et le nombre d'éléments désirés

```
int  CopyBuffer(
   int       indicator_handle,     // handle de l'indicateur
   int       buffer_num,           // numéro du buffer de l'indicateur
   int       start_pos,            // position de départ
   int       count,                // quantité à copier
   double    buffer[]              // tableau de destination pour la copie
   );
```

Appel avec la date de début et le nombre d'éléments désirés

```
int  CopyBuffer(
   int       indicator_handle,     // handle de l'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
   int       buffer_num,          // numéro du buffer de l'indicateur
   datetime  start_time,          // date et heure de début
   int       count,               // quantité à copier
   double    buffer[]             // tableau de destination pour la copie
   );
```

Appel avec les dates de début et de fin d'un intervalle de temps désiré

```
int  CopyBuffer(
   int       indicator_handle,    // handle de l'indicateur
   int       buffer_num,          // numéro du buffer de l'indicateur
   datetime  start_time,          // date et heure de début
   datetime  stop_time,           // date et heure de fin
   double    buffer[]             // tableau de destination pour la copie
   );
```

Parameters

indicator_handle

[in] Le handle sur l'indicateur, retourné par la fonction correspondante de l'indicateur.

buffer_num

[in] Le numéro du buffer de l'indicateur.

start_pos

[in] La position du premier élément à copier.

count

[in] Quantité de données à copier.

start_time

[in] Heure de la barre correspondant au premier élément.

stop_time

[in] Heure de la barre correspondant au dernier élément.

buffer[]

[out] Tableau d'éléments de type double.

Valeur de Retour

Retourne le nombre de données copiées ou -1 en cas d'erreur.

Note

Lors de la demande de données par un indicateur, si les timeseries demandées ne sont pas encore construites ou si elles doivent être téléchargées du serveur, la fonction retournera immédiatement -1, et le téléchargement ou la construction sera initiée.

Lors de la demande de données par un Expert Advisor ou un script, le téléchargement depuis le serveur sera lancé si le terminal n'a pas encore ces données localement, ou la construction des timeseries demandées sera lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction retournera la quantité de données qui seront prêtes au moment de l'expiration du délai.

© 2000-2024, MetaQuotes Ltd.

Exemple :

```
//+------------------------------------------------------------------+
//|                                                TestCopyBuffer3.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- dessine la MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- paramètres d'entrée
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE  price=PRICE_CLOSE;
input int                 shift=0;
//--- buffers de l'indicateur
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------+
//| Fonction d'initialisation d'un indicateur personnalisé           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Paramètre AsSeries = ",AsSeries);
   Print("Le buffer de l'indicateur après SetIndexBuffer() est une timeserie = ",
         ArrayGetAsSeries(MABuffer));
//--- définit le nom court de l'indicateur
   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- définit AsSeries (dépend des paramètres d'entrée)
   ArraySetAsSeries(MABuffer,AsSeries);
   Print("Le buffer de l'indicateur après ArraySetAsSeries(MABuffer,true); est une tir
         ArrayGetAsSeries(MABuffer));
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- vérifie si toutes les données sont calculées
   if(BarsCalculated(ma_handle)<rates_total) return(0);
//--- toutes les données ne peuvent pas être copiées
   int to_copy;
   if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
   else
     {
      to_copy=rates_total-prev_calculated;
      //--- la dernière valeur est toujours copiée
      to_copy++;
     }
//--- essaie de copier
   if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
//--- retourne la valeur de prev_calculated pour l'appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

L'exemple ci-dessus illustre comment le buffer d'un indicateur est rempli avec les valeurs du buffer d'un autre indicateur depuis l'indicateur sur le même symbole/la même période.

Un exemple détaillé de la demande de données historiques est visible dans la section Méthodes de Liaison d'Objet. Le script disponible dans cette section montre comment obtenir les valeurs de l'indicateur iFractals sur les 1000 dernières barres et comment afficher les 10 fractales haussières et baissières sur le graphique. Une technique identique peut être utilisée pour tous les indicateurs qui ont des données manquantes et qui sont habituellement dessinés avec les styles suivants :

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

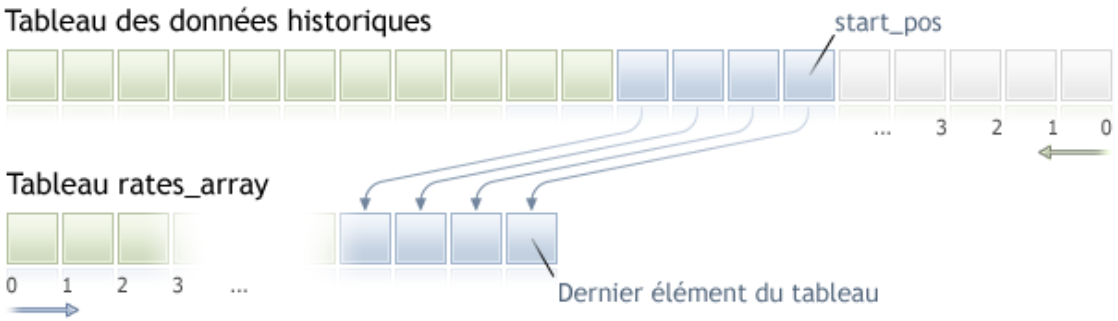
Voir aussi

Propriétés des Indicateurs Personnalisés, SetIndexBuffer

© 2000-2024, MetaQuotes Ltd.

CopyRates

Retourne dans le tableau rates_array les données historiques de la structure MqlRates pour un symbole et une période spécifiés. L'ordre des éléments des données copiées est du présent au passé, c'est à dire que la position de départ de 0 signifie la barre courante.



Lors de la copie d'une quantité inconnue de données, il est recommandé d'utiliser un tableau dynamique comme tableau destination, car si le nombre de données est inférieur (ou supérieur) que la longueur du tableau destination, la fonction essaye de réallouer la mémoire pour que l'ensemble des données demandées puisse être contenu.

Si vous connaissez la quantité de données à copier, il est préférable d'utiliser un buffer alloué de façon statique, afin d'éviter une allocation mémoire excessive.

Ceci quelque soit la propriété du tableau destination - as_series=true ou as_series=false. Les données seront copiées afin que le plus vieil élément soit situé au départ de la mémoire physique allouée pour le tableau. Il existe 3 variantes d'appel à la fonction.

Appel avec la première position et le nombre d'éléments désirés

```
int  CopyRates(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   int              start_pos,         // position de départ
   int              count,             // quantité de données à copier
   MqlRates         rates_array[]      // tableau de destination pour la copie
   );
```

Appel avec la date de début et le nombre d'éléments désirés

```
int  CopyRates(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   datetime         start_time,        // date et heure de début
   int              count,             // quantité de données à copier
   MqlRates         rates_array[]      // tableau de destination pour la copie
   );
```

Appel avec les dates de début et de fin d'un intervalle de temps désiré

```
int  CopyRates(
```

© 2000-2024, MetaQuotes Ltd.

```
   string          symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES timeframe,         // période
   datetime        start_time,        // date et heure de début
   datetime        stop_time,         // date et heure de fin
   MqlRates        rates_array[]      // tableau de destination pour la copie
   );
```

Parameters

symbol_name

[in] Nom du symbole.

timeframe

[in] Période.

start_time

[in] Heure de la barre correspondant au premier élément à copier.

start_pos

[in] La position de départ correspondant au 1er élément à copier.

count

[in] Quantité de données à copier.

stop_time

[in] Heure de la barre correspondant au dernier élément à copier.

rates_array[]

[out] Tableau d'éléments de type MqlRates.

Valeur de Retour

Retourne le nombre d'éléments copiés ou -1 en cas d'erreur.

Note

Si l'ensemble de l'intervalle des données demandées est en dehors des données disponibles sur le serveur, la fonction retourne -1. Si des données en dehors de TERMINAL_MAXBARS (nombre maximum de barres sur le graphique) sont demandées, la fonction retournera également -1.

Lors de la demande de données par un indicateur, si les timeseries demandées ne sont pas encore construites ou si elles doivent être téléchargées du serveur, la fonction retournera immédiatement -1, et le téléchargement ou la construction sera initiée.

Lors de la demande de données par un Expert Advisor ou un script, le téléchargement depuis le serveur sera lancé si le terminal n'a pas encore ces données localement, ou la construction des timeseries demandées sera lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction retournera la quantité de données qui seront prêtes au moment de l'expiration du délai, mais le téléchargement de l'historique continuera, et la fonction retournera plus de données à la prochaine demande similaire.

Lors de la demande de données à partir de la date de départ et du nombre désiré d'elements, seules les données dont la date est inférieure (donnée plus récente) ou égale à la date spécifiée seront retournées. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur

© 2000-2024, MetaQuotes Ltd.

est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours inférieure ou égale à celle spécifiée.

Lors de la demande de données dans un intervalle spécifié de dates, seules les données de cet intervalle seront retournées. L'intervalle est défini et compté en secondes. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours dans l'intervalle demandé.

Donc, si le jour courant est samedi, au moment d'essayer de copier les données sur l'intervalle de la semaine en spécifiant *start_time=Last_Tuesday* et *stop_time=Last_Friday*, la fonction retournera 0, car l'heure d'ouverture sur l'intervalle de la semaine est le dimanche, mais une barre 1 semaine ne rentre pas dans l'intervalle spécifié.

Si vous devez retourner une valeur correspondant à la barre non complète actuelle, vous pouvez utiliser la 1ère forme d'appel en spécifiant *start_pos*=0 et *count*=1.

Exemple :

```
void OnStart()
  {
//---
   MqlRates rates[];
   ArraySetAsSeries(rates,true);
   int copied=CopyRates(Symbol(),0,0,100,rates);
   if(copied>0)
     {
      Print("Barres copiées : "+copied);
      string format="open = %G, high = %G, low = %G, close = %G, volume = %d";
      string out;
      int size=fmin(copied,10);
      for(int i=0;i<size;i++)
        {
         out=i+":"+TimeToString(rates[i].time);
         out=out+" "+StringFormat(format,
                                  rates[i].open,
                                  rates[i].high,
                                  rates[i].low,
                                  rates[i].close,
                                  rates[i].tick_volume);
         Print(out);
        }
     }
   else Print("Echec de l'obtention des données historiques pour le symbole ",Symbol()
  }
```

Vous pouvez voir un exemple détaillé de demande des données historiques dans la section Méthodes de Liaison d'Objet. Le script disponible dans cette section montre comment obtenir les valeurs de l'indicateur iFractals sur les 1000 dernières barres et comment afficher les 10 fractales haussières et baissières sur le graphique. Une technique identique peut être utilisée pour tous les indicateurs qui ont des données manquantes et qui sont habituellement dessinés avec les styles suivants :

© 2000-2024, MetaQuotes Ltd.

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

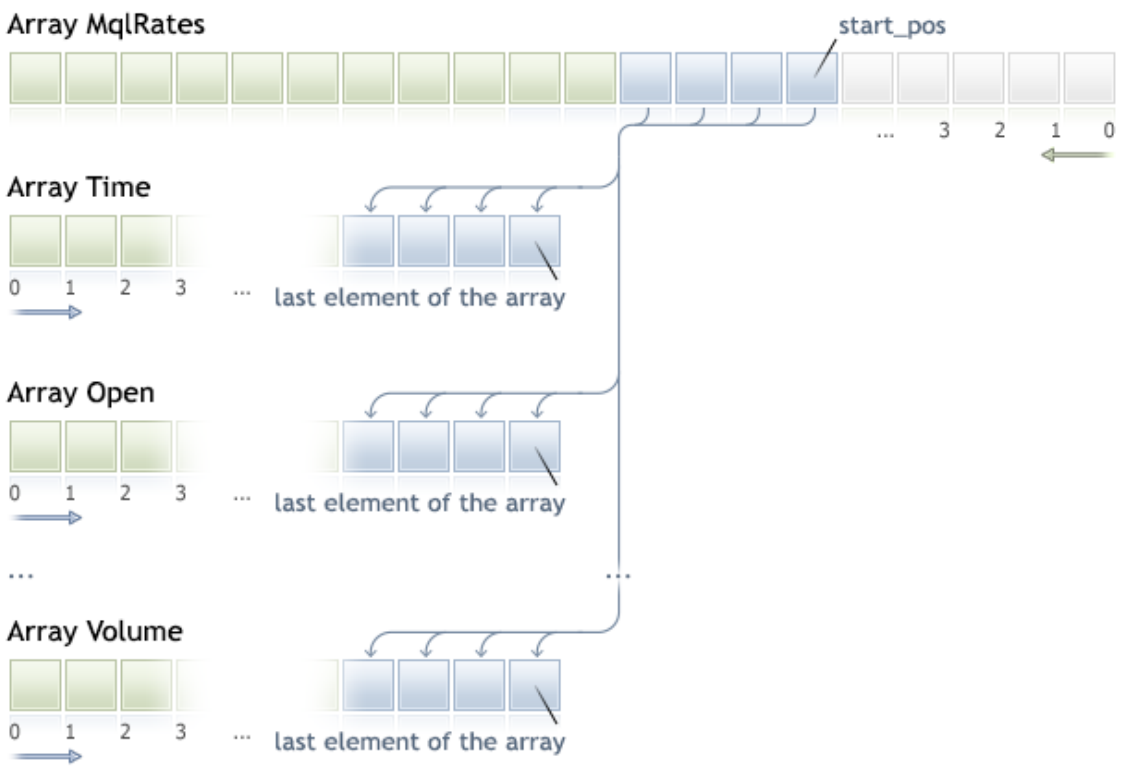
Voir aussi

Structures et Classes, TimeToString, StringFormat

© 2000-2024, MetaQuotes Ltd.

CopySeries

Retourne les timseries chronologiques de la structure MqlRates pour le symbole et la période spécifiés et pour la quantité demandée. Les données sont reçues dans les tableaux spécifiés. Les éléments sont comptés du présent vers le passé, ce qui signifie que la position de départ égale à 0 signifie la barre actuelle.



Si la quantité de données à copier est inconnue, il est recommandé d'utiliser un tableau dynamique pour les tableaux destination, car si la quantité de données dépasse ce qu'un tableau peut contenir, cela peut entraîner la tentative de redistribution du tableau pour qu'il corresponde à toutes les données demandées.

Si vous avez besoin de copier une quantité prédéterminée de données, il est recommandé d'utiliser un buffer alloué statiquement pour éviter une réallocation de mémoire inutile.

La propriété du tableau destination — as_series=true ou as_series=false — sera ignorée : lors de la copie, l'élément de la série chronologique le plus ancien sera copié au début de la mémoire physique allouée au tableau.

```
int  CopySeries(
   string            symbol_name,        // nom du symbole
   ENUM_TIMEFRAMES   timeframe,          // période
   int               start_pos,          // position de départ
   int               count,              // quantité d'éléments à copier
   ulong             rates_mask,         // combinaison de flags pour spécifier les sér
   void&             array1[],           // tableau destination de la première timeserie
   void&             array2[]            // tableau destination de la deuxième timeserie
```

© 2000-2024, MetaQuotes Ltd.

```
   ...
   );
```

Paramètres

symbol_name

[in] Symbole.

timeframe

[in] Période.

start_pos

[in] Indice du premier élément copié.

count

[in] Nombre d'éléments copiés.

rates_mask

[in] une combinaison de flags de l'énumération ENUM_COPY_RATES.

array1, array2,...

[out] Tableaux destination des séries chronologiques d'éléments de type MqlRates. L'ordre des tableaux passés à la fonction doit correspondre à l'ordre des champs de la structure MqlRates.

Valeur de Retour

Le nombre d'éléments copiés ou -1 si une erreur se produit.

Note

Si l'intervalle entier des données demandées est au-delà des données disponibles sur le serveur, la fonction renvoie -1. Si les données demandées dépassent TERMINAL_MAXBARS (le nombre maximum de barres sur le graphique), la fonction renvoie également -1.

Lors de la demande de données à partir d'un indicateur, la fonction renvoie immédiatement -1 si les séries temporelles demandées ne sont pas encore construites ou si elles doivent être téléchargées depuis le serveur. Mais le téléchargement/la construction des données sera lancé.

Lorsque vous demandez des données à un Expert Advisor ou à un script, le téléchargement depuis le serveur est lancé si le terminal ne dispose pas localement des données correspondantes, ou la construction des données nécessaires est lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction renvoie la quantité de données prêtes à l'expiration du délai d'attente. Mais le téléchargement de l'historique se poursuit et la fonction retourera plus de données lors de la prochaine requête similaire.

Différence entre CopySeries et CopyRates

La fonction CopySeries permet d'obtenir uniquement les séries temporelles nécessaires dans différents tableaux spécifiés au cours d'un appel, et toutes les données des séries temporelles seront synchronisées. Cela signifie que toutes les valeurs des tableaux destination à un certain indice N appartiendront à la même barre sur la paire Symbole/Timeframe spécifiée. Le développeur n'a donc pas besoin d'assurer la synchronisation de toutes les séries temporelles reçues par l'heure d'ouverture de la barre.

© 2000-2024, MetaQuotes Ltd.

Contrairement à CopyRates, qui renvoie l'ensemble complet des séries temporelles sous forme de tableau MqlRates, la fonction CopySeries permet au programmeur d'obtenir uniquement les séries temporelles requises sous forme de tableaux séparés. Cela peut être fait en spécifiant une combinaison d'indicateurs pour sélectionner le type de séries chronologiques. L'ordre des tableaux passés à la fonction doit correspondre à l'ordre des champs dans la structure MqlRates :

```
struct MqlRates
  {
   datetime time;         // heure de début
   double   open;         // prix d'ouverture
   double   high;         // prix le plus haut pour la période
   double   low;          // prix le plus bas pour la période
   double   close;        // prix de clôture
   long     tick_volume;  // volume du tick
   int      spread;       // spread
   long     real_volume;  // volume d'échange
  }
```

Donc, si vous avez besoin d'obtenir les séries chronologiques des valeurs *time*, *close* et *real_volume* pour les 100 dernières barres du symbole/période actuels, vous devez utiliser l'appel suivant :

```
datetime time[];
double   close[];
long     volume[];
CopySeries(NULL,0,0,100,COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_VOLUME_REAL,time,c
```

Faites bien attention à l'ordre des tableaux "*time*, *close*, *volume*" — il doit correspondre à l'ordre des champs dans la stucture MqlRates. L'ordre des valeurs dans *rates_mask* n'est pas important. Le masque pourrait être le suivant :

```
COPY_RATES_VOLUME_REAL|COPY_RATES_TIME|COPY_RATES_CLOSE
```

Exemple :

```
//--- paramètres d'entrée
input datetime InpDateFrom=D'2022.01.01 00:00:00';
input datetime InpDateTo  =D'2023.01.01 00:00:00';
input uint     InpCount   =20;
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- tableaux destination des timeseries de la structure de prix MqlRates
   double   open[];
   double   close[];
   float    closef[];
   datetime time1[], time2[];
//--- demande des prîx de clôture dans un tableau d'éléments de type double
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
   int res1=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE, time1, close);
   PrintFormat("1. CopySeries a retourné %d valeurs. Code d'erreur=%d", res1, GetLastE
   ArrayPrint(close);

//--- demande maintenant les prix d'ouverture ; utilise un tableau de float pour les p
   ResetLastError();
   int res2=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_OPEN, time2, open,
   PrintFormat("2. CopySeries a retourné %d valeurs. Code d'erreur=%d", res2, GetLastE
   ArrayPrint(closef);
//--- Compare les données reçues
   if((res1==res2) && (time1[0]==time2[0]))
     {
      Print("  | Time              |    Open       | Close double | Close float |");
      for(int i=0; i<10; i++)
        {
         PrintFormat("%d | %s |   %.5f    |   %.5f    |   %.5f   |",
                     i, TimeToString(time1[i]), open[i], close[i], closef[i]);
        }
     }
//--- Résultat
      1. CopySeries a retourné 20 valeurs. Code d'erreur=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684 1.(
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239 1.(
      2. CopySeries a retourné 20 valeurs. Code d'erreur=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684 1.(
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239 1.(
        | Time              |    Open       | Close double | Close float |
      0 | 2023.03.01 17:00 |   1.06660    |   1.06722    |   1.06722   |
      1 | 2023.03.01 18:00 |   1.06722    |   1.06733    |   1.06733   |
      2 | 2023.03.01 19:00 |   1.06734    |   1.06653    |   1.06653   |
      S3 | 2023.03.01 20:00 |   1.06654    |   1.06520    |   1.06520   |
      S4 | 2023.03.01 21:00 |   1.06520    |   1.06573    |   1.06573   |
      5 | 2023.03.01 22:00 |   1.06572    |   1.06649    |   1.06649   |
      6 | 2023.03.01 23:00 |   1.06649    |   1.06694    |   1.06694   |
      7 | 2023.03.02 00:00 |   1.06683    |   1.06675    |   1.06675   |
      8 | 2023.03.02 01:00 |   1.06675    |   1.06684    |   1.06684   |
      9 | 2023.03.02 02:00 |   1.06687    |   1.06604    |   1.06604   |
//---
  }
```

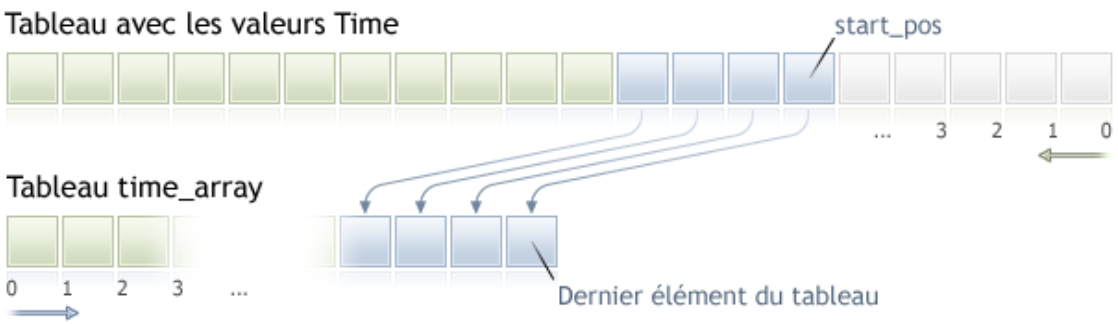
Voir aussi

Structures et classes, CopyRates

© 2000-2024, MetaQuotes Ltd.

CopyTime

La fonction copie la quantité spécifiée de données historiques d'heures d'ouverture des barres dans le paramètre time_array pour la paire symbole-période sélectionnée. Il est à noter que l'ordre des éléments est du présent vers le passé, c'est à dire que la position de départ de 0 signifie la barre courante.



Lors de la copie d'une quantité inconnue de données, il est recommandé d'utiliser un tableau dynamique comme tableau destination, car si le nombre de données est inférieur (ou supérieur) que la longueur du tableau destination, la fonction essaye de réallouer la mémoire pour que l'ensemble des données demandées puisse être contenu.

Si vous connaissez la quantité de données à copier, il est préférable d'utiliser un buffer alloué de façon statique, afin d'éviter une allocation mémoire excessive.

Ceci quelque soit la propriété du tableau destination - as_series=true ou as_series=false. Les données seront copiées afin que le plus vieil élément soit situé au départ de la mémoire physique allouée pour le tableau. Il existe 3 variantes d'appel à la fonction.

Appel avec la première position et le nombre d'éléments désirés

```
int  CopyTime(
   string            symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES   timeframe,        // période
   int               start_pos,        // position de départ
   int               count,            // quantité de données à copier
   datetime          time_array[]      // tableau de destination pour la copie open time
   );
```

Appel avec la date de début et le nombre d'éléments désirés

```
int  CopyTime(
   string            symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES   timeframe,        // période
   datetime          start_time,       // date et heure de début
   int               count,            // quantité de données à copier
   datetime          time_array[]      // tableau de destination pour la copie  open tir
   );
```

Appel avec les dates de début et de fin d'un intervalle de temps désiré

© 2000-2024, MetaQuotes Ltd.

```
int  CopyTime(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  timeframe,        // période
   datetime         start_time,       // date et heure de début
   datetime         stop_time,        // date et heure de fin
   datetime         time_array[]      // tableau de destination pour la copie open time
   );
```

Parameters

symbol_name

[in] Nom du symbole.

timeframe

[in] Période.

start_pos

[in] La position de départ correspondant au 1er élément à copier.

count

[in] Quantité de données à copier.

start_time

[in] Heure de la barre correspondant au premier élément à copier.

stop_time

[in] Heure de la barre correspondant au dernier élément à copier.

time_array[]

[out] Tableau d'éléments de type datetime.

Valeur de Retour

Retourne le nombre de données copiées ou -1 en cas d'erreur.

Note

Si l'ensemble de l'intervalle des données demandées est en dehors des données disponibles sur le serveur, la fonction retourne -1. Si des données en dehors de TERMINAL_MAXBARS (nombre maximum de barres sur le graphique) sont demandées, la fonction retournera également -1.

Lors de la demande de données par un indicateur, si les timeseries demandées ne sont pas encore construites ou si elles doivent être téléchargées du serveur, la fonction retournera immédiatement -1, et le téléchargement ou la construction sera initiée.

Lors de la demande de données par un Expert Advisor ou un script, le téléchargement depuis le serveur sera lancé si le terminal n'a pas encore ces données localement, ou la construction des timeseries demandées sera lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction retournera la quantité de données qui seront prêtes au moment de l'expiration du délai, mais le téléchargement de l'historique continuera, et la fonction retournera plus de données à la prochaine demande similaire.

Lors de la demande de données à partir de la date de départ et du nombre désiré d'elements, seules les données dont la date est inférieure (donnée plus récente) ou égale à la date spécifiée seront

© 2000-2024, MetaQuotes Ltd.

retournées. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours inférieure ou égale à celle spécifiée.

Lors de la demande de données dans un intervalle spécifié de dates, seules les données de cet intervalle seront retournées. L'intervalle est défini et compté en secondes. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours dans l'intervalle demandé.

Donc, si le jour courant est samedi, au moment d'essayer de copier les données sur l'intervalle de la semaine en spécifiant *start_time=Last_Tuesday* et *stop_time=Last_Friday*, la fonction retournera 0, car l'heure d'ouverture sur l'intervalle de la semaine est le dimanche, mais une barre 1 semaine ne rentre pas dans l'intervalle spécifié.

Si vous devez retourner une valeur correspondant à la barre non complète actuelle, vous pouvez utiliser la 1ère forme d'appel en spécifiant *start_pos*=0 et *count*=1.

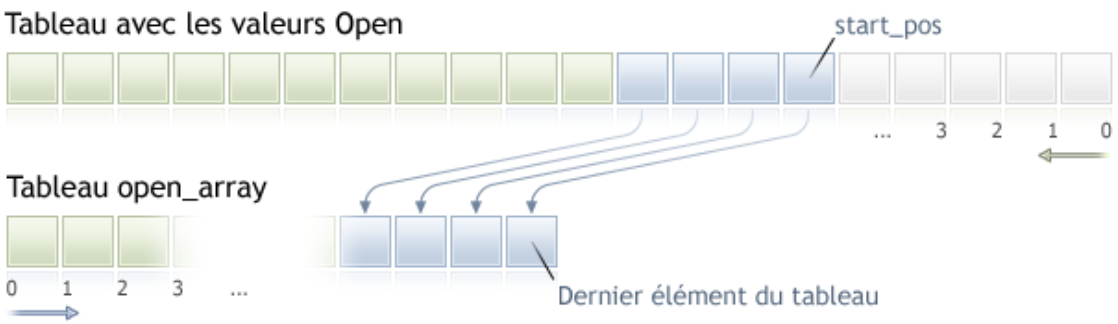
Vous pouvez voir un exemple détaillé de demande des données historiques dans la section Méthodes de Liaison d'Objet. Le script disponible dans cette section montre comment obtenir les valeurs de l'indicateur iFractals sur les 1000 dernières barres et comment afficher les 10 fractales haussières et baissières sur le graphique. Une technique identique peut être utilisée pour tous les indicateurs qui ont des données manquantes et qui sont habituellement dessinés avec les styles suivants :

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyOpen

La fonction copie la quantité spécifiée de données historiques des prix d'ouverture des barres dans le paramètre open_array pour la paire symbole-période sélectionnée. Il est à noter que l'ordre des éléments est du présent vers le passé, c'est à dire que la position de départ de 0 signifie la barre courante.



Lors de la copie d'une quantité inconnue de données, il est recommandé d'utiliser un tableau dynamique comme tableau destination, car si le nombre de données est inférieur (ou supérieur) que la longueur du tableau destination, la fonction essaye de réallouer la mémoire pour que l'ensemble des données demandées puisse être contenu.

Si vous connaissez la quantité de données à copier, il est préférable d'utiliser un buffer alloué de façon statique, afin d'éviter une allocation mémoire excessive.

Ceci quelque soit la propriété du tableau destination - as_series=true ou as_series=false. Les données seront copiées afin que le plus vieil élément soit situé au départ de la mémoire physique allouée pour le tableau. Il existe 3 variantes d'appel à la fonction.

Appel avec la première position et le nombre d'éléments désirés

```
int  CopyOpen(
   string           symbol_name,     // nom du symbole
   ENUM_TIMEFRAMES  timeframe,       // période
   int              start_pos,       // position de départ
   int              count,           // quantité de données à copier
   double           open_array[]     // tableau de destination pour la copie open pric
   );
```

Appel avec la date de début et le nombre d'éléments désirés

```
int  CopyOpen(
   string           symbol_name,     // nom du symbole
   ENUM_TIMEFRAMES  timeframe,       // période
   datetime         start_time,      // date et heure de début
   int              count,           // quantité de données à copier
   double           open_array[]     // tableau destination pour les prix d'ouverture
   );
```

Appel avec les dates de début et de fin d'un intervalle de temps désiré

© 2000-2024, MetaQuotes Ltd.

```
int  CopyOpen(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  timeframe,        // période
   datetime         start_time,       // date et heure de début
   datetime         stop_time,        // date et heure de fin
   double           open_array[]      // tableau destination pour les prix d'ouverture
   );
```

Parameters

symbol_name

[in] Nom du symbole.

timeframe

[in] Période.

start_pos

[in] La position de départ correspondant au 1er élément à copier.

count

[in] Quantité de données à copier.

start_time

[in] Heure de la barre correspondant au premier élément à copier.

stop_time

[in] Heure de la barre correspondant au dernier élément à copier.

open_array[]

[out] Tableau d'éléments de type double.

Valeur de Retour

Retourne le nombre d'éléments dans le tableau ou -1 en cas d'erreur.

Note

Si l'ensemble de l'intervalle des données demandées est en dehors des données disponibles sur le serveur, la fonction retourne -1. Si des données en dehors de TERMINAL_MAXBARS (nombre maximum de barres sur le graphique) sont demandées, la fonction retournera également -1.

Lors de la demande de données par un indicateur, si les timeseries demandées ne sont pas encore construites ou si elles doivent être téléchargées du serveur, la fonction retournera immédiatement -1, et le téléchargement ou la construction sera initiée.

Lors de la demande de données par un Expert Advisor ou un script, le téléchargement depuis le serveur sera lancé si le terminal n'a pas encore ces données localement, ou la construction des timeseries demandées sera lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction retournera la quantité de données qui seront prêtes au moment de l'expiration du délai, mais le téléchargement de l'historique continuera, et la fonction retournera plus de données à la prochaine demande similaire.

Lors de la demande de données à partir de la date de départ et du nombre désiré d'elements, seules les données dont la date est inférieure (donnée plus récente) ou égale à la date spécifiée seront

© 2000-2024, MetaQuotes Ltd.

retournées. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours inférieure ou égale à celle spécifiée.

Lors de la demande de données dans un intervalle spécifié de dates, seules les données de cet intervalle seront retournées. L'intervalle est défini et compté en secondes. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours dans l'intervalle demandé.

Donc, si le jour courant est samedi, au moment d'essayer de copier les données sur l'intervalle de la semaine en spécifiant *start_time=Last_Tuesday* et *stop_time=Last_Friday*, la fonction retournera 0, car l'heure d'ouverture sur l'intervalle de la semaine est le dimanche, mais une barre 1 semaine ne rentre pas dans l'intervalle spécifié.

Si vous devez retourner une valeur correspondant à la barre non complète actuelle, vous pouvez utiliser la 1ère forme d'appel en spécifiant *start_pos*=0 et *count*=1.

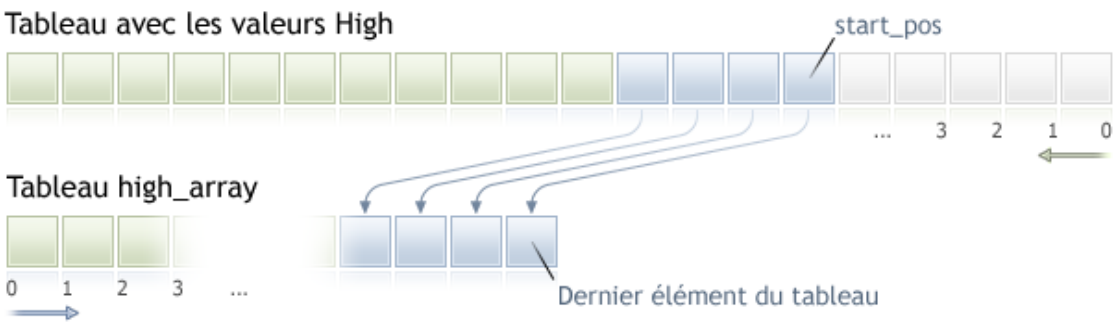
Vous pouvez voir un exemple détaillé de demande des données historiques dans la section Méthodes de Liaison d'Objet. Le script disponible dans cette section montre comment obtenir les valeurs de l'indicateur iFractals sur les 1000 dernières barres et comment afficher les 10 fractales haussières et baissières sur le graphique. Une technique identique peut être utilisée pour tous les indicateurs qui ont des données manquantes et qui sont habituellement dessinés avec les styles suivants :

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyHigh

La fonction copie la quantité spécifiée de données historiques des prix les plus hauts des barres dans le paramètre high_array pour la paire symbole-période sélectionnée. Il est à noter que l'ordre des éléments est du présent vers le passé, c'est à dire que la position de départ de 0 signifie la barre courante.



Lors de la copie d'une quantité inconnue de données, il est recommandé d'utiliser un tableau dynamique comme tableau destination, car si le nombre de données est inférieur (ou supérieur) que la longueur du tableau destination, la fonction essaye de réallouer la mémoire pour que l'ensemble des données demandées puisse être contenu.

Si vous connaissez la quantité de données à copier, il est préférable d'utiliser un buffer alloué de façon statique, afin d'éviter une allocation mémoire excessive.

Ceci quelque soit la propriété du tableau destination - as_series=true ou as_series=false. Les données seront copiées afin que le plus vieil élément soit situé au départ de la mémoire physique allouée pour le tableau. Il existe 3 variantes d'appel à la fonction.

Appel avec la première position et le nombre d'éléments désirés

```
int  CopyHigh(
   string            symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES   timeframe,         // période
   int               start_pos,         // position de départ
   int               count,             // quantité de données à copier
   double            high_array[]       // tableau de destination pour la copie
   );
```

Appel avec la date de début et le nombre d'éléments désirés

```
int  CopyHigh(
   string            symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES   timeframe,         // période
   datetime          start_time,        // date et heure de début
   int               count,             // quantité de données à copier
   double            high_array[]       // tableau de destination pour la copie
   );
```

Appel avec les dates de début et de fin d'un intervalle de temps désiré

© 2000-2024, MetaQuotes Ltd.

```
int  CopyHigh(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  timeframe,        // période
   datetime         start_time,       // date et heure de début
   datetime         stop_time,        // date et heure de fin
   double           high_array[]      // tableau de destination pour la copie
   );
```

Parameters

symbol_name

[in] Nom du symbole.

timeframe

[in] Période.

start_pos

[in] La position de départ correspondant au 1er élément à copier.

count

[in] Quantité de données à copier.

start_time

[in] Heure de la barre correspondant au premier élément à copier.

stop_time

[in] Heure de la barre correspondant au dernier élément à copier.

high_array[]

[out] Tableau d'éléments de type double.

Valeur de Retour

Retourne le nombre de données copiées ou -1 en cas d'erreur.

Note

Si l'ensemble de l'intervalle des données demandées est en dehors des données disponibles sur le serveur, la fonction retourne -1. Si des données en dehors de TERMINAL_MAXBARS (nombre maximum de barres sur le graphique) sont demandées, la fonction retournera également -1.

Lors de la demande de données par un indicateur, si les timeseries demandées ne sont pas encore construites ou si elles doivent être téléchargées du serveur, la fonction retournera immédiatement -1, et le téléchargement ou la construction sera initiée.

Lors de la demande de données par un Expert Advisor ou un script, le téléchargement depuis le serveur sera lancé si le terminal n'a pas encore ces données localement, ou la construction des timeseries demandées sera lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction retournera la quantité de données qui seront prêtes au moment de l'expiration du délai, mais le téléchargement de l'historique continuera, et la fonction retournera plus de données à la prochaine demande similaire.

Lors de la demande de données à partir de la date de départ et du nombre désiré d'elements, seules les données dont la date est inférieure (donnée plus récente) ou égale à la date spécifiée seront

© 2000-2024, MetaQuotes Ltd.

retournées. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours inférieure ou égale à celle spécifiée.

Lors de la demande de données dans un intervalle spécifié de dates, seules les données de cet intervalle seront retournées. L'intervalle est défini et compté en secondes. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours dans l'intervalle demandé.

Donc, si le jour courant est samedi, au moment d'essayer de copier les données sur l'intervalle de la semaine en spécifiant *start_time=Last_Tuesday* et *stop_time=Last_Friday*, la fonction retournera 0, car l'heure d'ouverture sur l'intervalle de la semaine est le dimanche, mais une barre 1 semaine ne rentre pas dans l'intervalle spécifié.

Si vous devez retourner une valeur correspondant à la barre non complète actuelle, vous pouvez utiliser la 1ère forme d'appel en spécifiant *start_pos*=0 et *count*=1.

Exemple :

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Exemple de sortie des prix High[i] et Low[i]"
#property description "pour une barre choisie aléatoirement"

double High[],Low[];
//+------------------------------------------------------------------+
//| Obtient le Low pour l'indice de barre spécifié                   |
//+------------------------------------------------------------------+
double iLow(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double low=0;
   ArraySetAsSeries(Low,true);
   int copied=CopyLow(symbol,timeframe,0,Bars(symbol,timeframe),Low);
   if(copied>0 && index<copied) low=Low[index];
   return(low);
  }
//+------------------------------------------------------------------+
//| Obtient le High pour l'indice de barre spécifié                  |
//+------------------------------------------------------------------+
double iHigh(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double high=0;
   ArraySetAsSeries(High,true);
   int copied=CopyHigh(symbol,timeframe,0,Bars(symbol,timeframe),High);
   if(copied>0 && index<copied) high=High[index];
   return(high);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Fonction de tick de l'Expert                                     |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- pour chaque tick, sort les valeurs High et Low de la barre avec l'index,
//--- qui est égal au second, sur lequel le tick est arrivé
   datetime t=TimeCurrent();
   int sec=t%60;
   printf("High[%d] = %G  Low[%d] = %G",
          sec,iHigh(Symbol(),0,sec),
          sec,iLow(Symbol(),0,sec));
  }
```

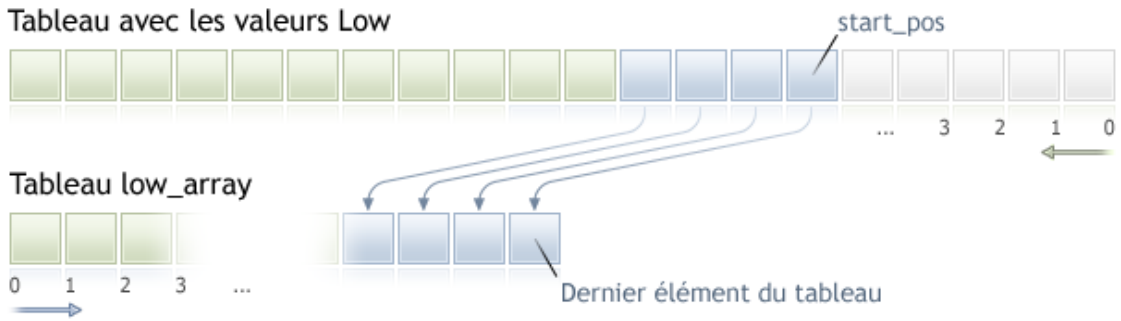
Vous pouvez voir un exemple détaillé de demande des données historiques dans la section Méthodes de Liaison d'Objet. Le script disponible dans cette section montre comment obtenir les valeurs de l'indicateur iFractals sur les 1000 dernières barres et comment afficher les 10 fractales haussières et baissières sur le graphique. Une technique identique peut être utilisée pour tous les indicateurs qui ont des données manquantes et qui sont habituellement dessinés avec les styles suivants :

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyLow

La fonction copie la quantité spécifiée de données historiques des prix les plus bas des barres dans le paramètre low_array pour la paire symbole-période sélectionnée. Il est à noter que l'ordre des éléments est du présent vers le passé, c'est à dire que la position de départ de 0 signifie la barre courante.



Lors de la copie d'une quantité inconnue de données, il est recommandé d'utiliser un tableau dynamique comme tableau destination, car si le nombre de données est inférieur (ou supérieur) que la longueur du tableau destination, la fonction essaye de réallouer la mémoire pour que l'ensemble des données demandées puisse être contenu.

Si vous connaissez la quantité de données à copier, il est préférable d'utiliser un buffer alloué de façon statique, afin d'éviter une allocation mémoire excessive.

Ceci quelque soit la propriété du tableau destination - as_series=true ou as_series=false. Les données seront copiées afin que le plus vieil élément soit situé au départ de la mémoire physique allouée pour le tableau. Il existe 3 variantes d'appel à la fonction.

Appel avec la première position et le nombre d'éléments désirés

```
int  CopyLow(
   string            symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES   timeframe,        // période
   int               start_pos,        // position de départ
   int               count,            // quantité de données à copier
   double            low_array[]       // tableau de destination pour la copie
   );
```

Appel avec la date de début et le nombre d'éléments désirés

```
int  CopyLow(
   string            symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES   timeframe,        // période
   datetime          start_time,       // date et heure de début
   int               count,            // quantité de données à copier
   double            low_array[]       // tableau de destination pour la copie
   );
```

Appel avec les dates de début et de fin d'un intervalle de temps désiré

```
int  CopyLow(
```

© 2000-2024, MetaQuotes Ltd.

```
   string          symbol_name,     // nom du symbole
   ENUM_TIMEFRAMES timeframe,       // période
   datetime        start_time,      // date et heure de début
   datetime        stop_time,       // date et heure de fin
   double          low_array[]      // tableau de destination pour la copie
   );
```

Parameters

symbol_name

[in] Symbole.

timeframe

[in] Période.

start_pos

[in] La position de départ correspondant au 1er élément à copier.

count

[in] Quantité de données à copier.

start_time

[in] Heure de la barre correspondant au premier élément à copier.

stop_time

[in] Heure de la barre correspondant au dernier élément à copier.

low_array[]

[out] Tableau d'éléments de type double.

Valeur de Retour

Retourne le nombre de données copiées ou -1 en cas d'erreur.

Note

Si l'ensemble de l'intervalle des données demandées est en dehors des données disponibles sur le serveur, la fonction retourne -1. Si des données en dehors de TERMINAL_MAXBARS (nombre maximum de barres sur le graphique) sont demandées, la fonction retournera également -1.

Lors de la demande de données par un indicateur, si les timeseries demandées ne sont pas encore construites ou si elles doivent être téléchargées du serveur, la fonction retournera immédiatement -1, et le téléchargement ou la construction sera initiée.

Lors de la demande de données par un Expert Advisor ou un script, le téléchargement depuis le serveur sera lancé si le terminal n'a pas encore ces données localement, ou la construction des timeseries demandées sera lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction retournera la quantité de données qui seront prêtes au moment de l'expiration du délai, mais le téléchargement de l'historique continuera, et la fonction retournera plus de données à la prochaine demande similaire.

Lors de la demande de données à partir de la date de départ et du nombre désiré d'elements, seules les données dont la date est inférieure (donnée plus récente) ou égale à la date spécifiée seront retournées. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur

© 2000-2024, MetaQuotes Ltd.

est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours inférieure ou égale à celle spécifiée.

Lors de la demande de données dans un intervalle spécifié de dates, seules les données de cet intervalle seront retournées. L'intervalle est défini et compté en secondes. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours dans l'intervalle demandé.

Donc, si le jour courant est samedi, au moment d'essayer de copier les données sur l'intervalle de la semaine en spécifiant *start_time=Last_Tuesday* et *stop_time=Last_Friday*, la fonction retournera 0, car l'heure d'ouverture sur l'intervalle de la semaine est le dimanche, mais une barre 1 semaine ne rentre pas dans l'intervalle spécifié.

Si vous devez retourner une valeur correspondant à la barre non complète actuelle, vous pouvez utiliser la 1ère forme d'appel en spécifiant *start_pos*=0 et *count*=1.

Vous pouvez voir un exemple détaillé de demande des données historiques dans la section Méthodes de Liaison d'Objet. Le script disponible dans cette section montre comment obtenir les valeurs de l'indicateur iFractals sur les 1000 dernières barres et comment afficher les 10 fractales haussières et baissières sur le graphique. Une technique identique peut être utilisée pour tous les indicateurs qui ont des données manquantes et qui sont habituellement dessinés avec les styles suivants :

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

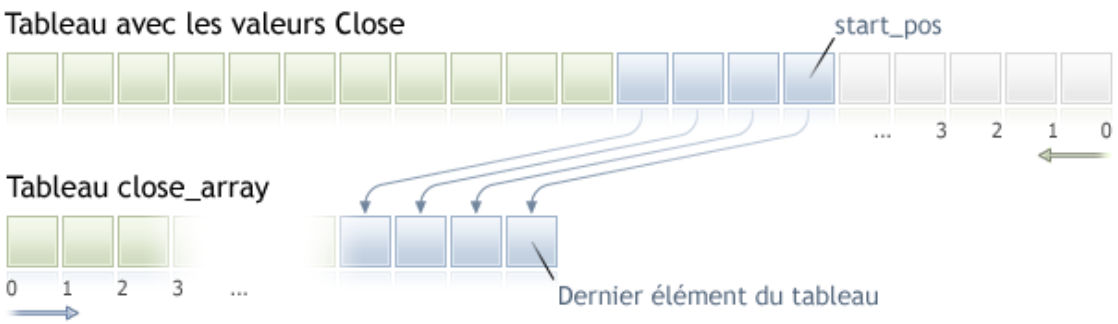
Voir aussi

CopyHigh

© 2000-2024, MetaQuotes Ltd.

CopyClose

La fonction copie la quantité spécifiée de données historiques des prix de clôture des barres dans le paramètre close_array pour la paire symbole-période sélectionnée. Il est à noter que l'ordre des éléments est du présent vers le passé, c'est à dire que la position de départ de 0 signifie la barre courante.



Lors de la copie d'une quantité inconnue de données, il est recommandé d'utiliser un tableau dynamique comme tableau destination, car si le nombre de données est inférieur (ou supérieur) que la longueur du tableau destination, la fonction essaye de réallouer la mémoire pour que l'ensemble des données demandées puisse être contenu.

Si vous connaissez la quantité de données à copier, il est préférable d'utiliser un buffer alloué de façon statique, afin d'éviter une allocation mémoire excessive.

Ceci quelque soit la propriété du tableau destination - as_series=true ou as_series=false. Les données seront copiées afin que le plus vieil élément soit situé au départ de la mémoire physique allouée pour le tableau. Il existe 3 variantes d'appel à la fonction.

Appel avec la première position et le nombre d'éléments désirés

```
int  CopyClose(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   int              start_pos,         // position de départ
   int              count,             // quantité de données à copier
   double           close_array[]      // tableau de destination pour la copie
   );
```

Appel avec la date de début et le nombre d'éléments désirés

```
int  CopyClose(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   datetime         start_time,        // date et heure de début
   int              count,             // quantité de données à copier
   double           close_array[]      // tableau de destination pour la copie
   );
```

Appel avec les dates de début et de fin d'un intervalle de temps désiré

© 2000-2024, MetaQuotes Ltd.

```
int  CopyClose(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   datetime         start_time,        // date et heure de début
   datetime         stop_time,         // date et heure de fin
   double           close_array[]      // tableau de destination pour la copie
   );
```

Parameters

symbol_name

[in] Nom du symbole.

timeframe

[in] Période.

start_pos

[in] La position de départ correspondant au 1er élément à copier.

count

[in] Quantité de données à copier.

start_time

[in] Heure de la barre correspondant au premier élément à copier.

stop_time

[in] Heure de la barre correspondant au dernier élément à copier.

close_array[]

[out] Tableau d'éléments de type double.

Valeur de Retour

Retourne le nombre de données copiées ou -1 en cas d'erreur.

Note

Si l'ensemble de l'intervalle des données demandées est en dehors des données disponibles sur le serveur, la fonction retourne -1. Si des données en dehors de TERMINAL_MAXBARS (nombre maximum de barres sur le graphique) sont demandées, la fonction retournera également -1.

Lors de la demande de données par un indicateur, si les timeseries demandées ne sont pas encore construites ou si elles doivent être téléchargées du serveur, la fonction retournera immédiatement -1, et le téléchargement ou la construction sera initiée.

Lors de la demande de données par un Expert Advisor ou un script, le téléchargement depuis le serveur sera lancé si le terminal n'a pas encore ces données localement, ou la construction des timeseries demandées sera lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction retournera la quantité de données qui seront prêtes au moment de l'expiration du délai, mais le téléchargement de l'historique continuera, et la fonction retournera plus de données à la prochaine demande similaire.

Lors de la demande de données à partir de la date de départ et du nombre désiré d'elements, seules les données dont la date est inférieure (donnée plus récente) ou égale à la date spécifiée seront

© 2000-2024, MetaQuotes Ltd.

retournées. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours inférieure ou égale à celle spécifiée.

Lors de la demande de données dans un intervalle spécifié de dates, seules les données de cet intervalle seront retournées. L'intervalle est défini et compté en secondes. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours dans l'intervalle demandé.

Donc, si le jour courant est samedi, au moment d'essayer de copier les données sur l'intervalle de la semaine en spécifiant *start_time=Last_Tuesday* et *stop_time=Last_Friday*, la fonction retournera 0, car l'heure d'ouverture sur l'intervalle de la semaine est le dimanche, mais une barre 1 semaine ne rentre pas dans l'intervalle spécifié.

Si vous devez retourner une valeur correspondant à la barre non complète actuelle, vous pouvez utiliser la 1ère forme d'appel en spécifiant *start_pos*=0 et *count*=1.

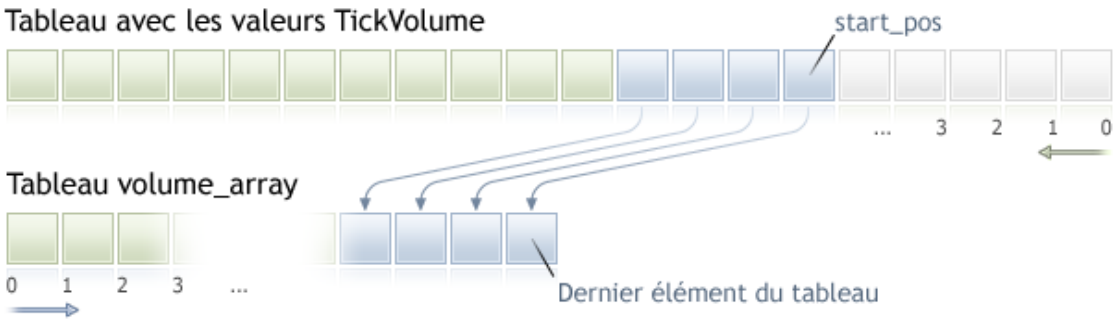
Vous pouvez voir un exemple détaillé de demande des données historiques dans la section Méthodes de Liaison d'Objet. Le script disponible dans cette section montre comment obtenir les valeurs de l'indicateur iFractals sur les 1000 dernières barres et comment afficher les 10 fractales haussières et baissières sur le graphique. Une technique identique peut être utilisée pour tous les indicateurs qui ont des données manquantes et qui sont habituellement dessinés avec les styles suivants :

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyTickVolume

La fonction copie la quantité spécifiée de données historiques du volume des ticks dans le paramètre volume_array pour la paire symbole-période sélectionnée. Il est à noter que l'ordre des éléments est du présent vers le passé, c'est à dire que la position de départ de 0 signifie la barre courante.



Lors de la copie d'une quantité inconnue de données, il est recommandé d'utiliser un tableau dynamique comme tableau destination, car si le nombre de données est inférieur (ou supérieur) que la longueur du tableau destination, la fonction essaye de réallouer la mémoire pour que l'ensemble des données demandées puisse être contenu.

Si vous connaissez la quantité de données à copier, il est préférable d'utiliser un buffer alloué de façon statique, afin d'éviter une allocation mémoire excessive.

Ceci quelque soit la propriété du tableau destination - as_series=true ou as_series=false. Les données seront copiées afin que le plus vieil élément soit situé au départ de la mémoire physique allouée pour le tableau. Il existe 3 variantes d'appel à la fonction.

Appel avec la première position et le nombre d'éléments désirés

```
int  CopyTickVolume(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   int              start_pos,         // position de départ
   int              count,             // quantité de données à copier
   long             volume_array[]     // tableau destination pour le volume des ticks
   );
```

Appel avec la date de début et le nombre d'éléments désirés

```
int  CopyTickVolume(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   datetime         start_time,        // date et heure de début
   int              count,             // quantité de données à copier
   long             volume_array[]     // tableau destination pour le volume des ticks
   );
```

Appel avec les dates de début et de fin d'un intervalle de temps désiré

```
int  CopyTickVolume(
```

© 2000-2024, MetaQuotes Ltd.

```
   string          symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES timeframe,        // période
   datetime        start_time,       // date et heure de début
   datetime        stop_time,        // date et heure de fin
   long            volume_array[]    // tableau destination pour le volume des ticks
   );
```

Parameters

symbol_name

[in] Nom du symbole.

timeframe

[in] Période.

start_pos

[in] La position de départ correspondant au 1er élément à copier.

count

[in] Quantité de données à copier.

start_time

[in] Heure de la barre correspondant au premier élément à copier.

stop_time

[in] Heure de la barre correspondant au dernier élément à copier.

volume_array[]

[out] Tableau d'éléments de type long.

Valeur de Retour

Retourne le nombre de données copiées ou -1 en cas d'erreur.

Note

Si l'ensemble de l'intervalle des données demandées est en dehors des données disponibles sur le serveur, la fonction retourne -1. Si des données en dehors de TERMINAL_MAXBARS (nombre maximum de barres sur le graphique) sont demandées, la fonction retournera également -1.

Lors de la demande de données par un indicateur, si les timeseries demandées ne sont pas encore construites ou si elles doivent être téléchargées du serveur, la fonction retournera immédiatement -1, et le téléchargement ou la construction sera initiée.

Lors de la demande de données par un Expert Advisor ou un script, le téléchargement depuis le serveur sera lancé si le terminal n'a pas encore ces données localement, ou la construction des timeseries demandées sera lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction retournera la quantité de données qui seront prêtes au moment de l'expiration du délai, mais le téléchargement de l'historique continuera, et la fonction retournera plus de données à la prochaine demande similaire.

Lors de la demande de données à partir de la date de départ et du nombre désiré d'elements, seules les données dont la date est inférieure (donnée plus récente) ou égale à la date spécifiée seront retournées. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur

© 2000-2024, MetaQuotes Ltd.

est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours inférieure ou égale à celle spécifiée.

Lors de la demande de données dans un intervalle spécifié de dates, seules les données de cet intervalle seront retournées. L'intervalle est défini et compté en secondes. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours dans l'intervalle demandé.

Donc, si le jour courant est samedi, au moment d'essayer de copier les données sur l'intervalle de la semaine en spécifiant *start_time=Last_Tuesday* et *stop_time=Last_Friday*, la fonction retournera 0, car l'heure d'ouverture sur l'intervalle de la semaine est le dimanche, mais une barre 1 semaine ne rentre pas dans l'intervalle spécifié.

Si vous devez retourner une valeur correspondant à la barre non complète actuelle, vous pouvez utiliser la 1ère forme d'appel en spécifiant *start_pos*=0 et *count*=1.

Exemple :

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- dessine le TickVolume
#property indicator_label1  "TickVolume"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  C'143,188,139'
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- paramètres d'entrée
input int      bars=3000;
//--- buffers de l'indicateur
double         TickVolumeBuffer[];
//+------------------------------------------------------------------+
//| Fonction d'initialisation d'un indicateur personnalisé           |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

© 2000-2024, MetaQuotes Ltd.

```
                  const double &close[],
                  const long &tick_volume[],
                  const long &volume[],
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      long timeseries[];
      ArraySetAsSeries(timeseries,true);
      int prices=CopyTickVolume(Symbol(),0,0,bars,timeseries);
      for(int i=0;i<rates_total-prices;i++) TickVolumeBuffer[i]=0.0;
      for(int i=0;i<prices;i++) TickVolumeBuffer[rates_total-1-i]=timeseries[prices-1-
      Print("Nombre de valeurs de TickVolume reçues : "+prices);
     }
   else
     {
      long timeseries[];
      int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
      TickVolumeBuffer[rates_total-1]=timeseries[0];
     }
//--- retourne la valeur de prev_calculated pour l'appel suivant
   return(rates_total);
  }
```

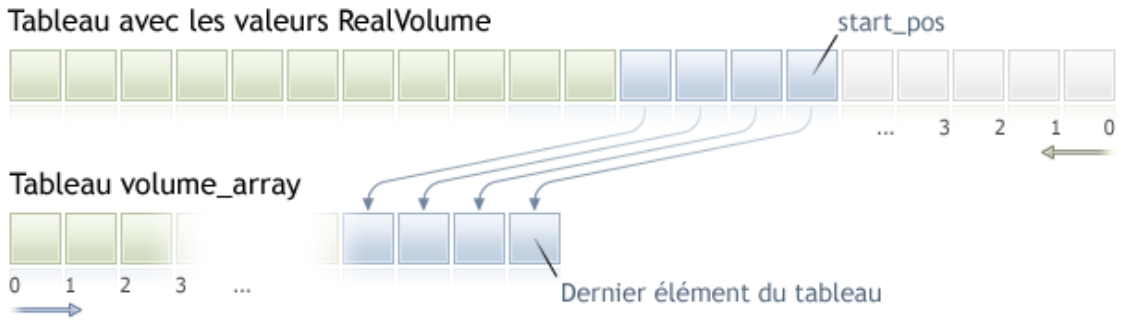
Vous pouvez voir un exemple détaillé de demande des données historiques dans la section Méthodes de Liaison d'Objet. Le script disponible dans cette section montre comment obtenir les valeurs de l'indicateur iFractals sur les 1000 dernières barres et comment afficher les 10 fractales haussières et baissières sur le graphique. Une technique identique peut être utilisée pour tous les indicateurs qui ont des données manquantes et qui sont habituellement dessinés avec les styles suivants :

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyRealVolume

La fonction copie la quantité spécifiée de données historiques du volume des trades dans le paramètre volume_array pour la paire symbole-période sélectionnée. Il est à noter que l'ordre des éléments est du présent vers le passé, c'est à dire que la position de départ de 0 signifie la barre courante.



Lors de la copie d'une quantité inconnue de données, il est recommandé d'utiliser un tableau dynamique comme tableau destination, car si le nombre de données est inférieur (ou supérieur) que la longueur du tableau destination, la fonction essaye de réallouer la mémoire pour que l'ensemble des données demandées puisse être contenu.

Si vous connaissez la quantité de données à copier, il est préférable d'utiliser un buffer alloué de façon statique, afin d'éviter une allocation mémoire excessive.

Ceci quelque soit la propriété du tableau destination - as_series=true ou as_series=false. Les données seront copiées afin que le plus vieil élément soit situé au départ de la mémoire physique allouée pour le tableau. Il existe 3 variantes d'appel à la fonction.

Appel avec la première position et le nombre d'éléments désirés

```
int  CopyRealVolume(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   int              start_pos,         // position de départ
   int              count,             // quantité de données à copier
   long             volume_array[]     // tableau destination pour les volumes
   );
```

Appel avec la date de début et le nombre d'éléments désirés

```
int  CopyRealVolume(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   datetime         start_time,        // date et heure de début
   int              count,             // quantité de données à copier
   long             volume_array[]     // tableau destination pour les volumes
   );
```

Appel avec les dates de début et de fin d'un intervalle de temps désiré

```
int  CopyRealVolume(
```

© 2000-2024, MetaQuotes Ltd.

```
   string          symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES timeframe,        // période
   datetime        start_time,       // date et heure de début
   datetime        stop_time,        // date et heure de fin
   long            volume_array[]    // tableau destination pour les volumes
   );
```

Parameters

symbol_name

[in] Nom du symbole.

timeframe

[in] Période.

start_pos

[in] La position de départ correspondant au 1er élément à copier.

count

[in] Quantité de données à copier.

start_time

[in] Heure de la barre correspondant au premier élément à copier.

stop_time

[in] Heure de la barre correspondant au dernier élément à copier.

volume_array[]

[out] Tableau d'éléments de type long.

Valeur de Retour

Retourne le nombre de données copiées ou -1 en cas d'erreur.

Note

Si l'ensemble de l'intervalle des données demandées est en dehors des données disponibles sur le serveur, la fonction retourne -1. Si des données en dehors de TERMINAL_MAXBARS (nombre maximum de barres sur le graphique) sont demandées, la fonction retournera également -1.

Lors de la demande de données par un indicateur, si les timeseries demandées ne sont pas encore construites ou si elles doivent être téléchargées du serveur, la fonction retournera immédiatement -1, et le téléchargement ou la construction sera initiée.

Lors de la demande de données par un Expert Advisor ou un script, le téléchargement depuis le serveur sera lancé si le terminal n'a pas encore ces données localement, ou la construction des timeseries demandées sera lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction retournera la quantité de données qui seront prêtes au moment de l'expiration du délai, mais le téléchargement de l'historique continuera, et la fonction retournera plus de données à la prochaine demande similaire.

Lors de la demande de données à partir de la date de départ et du nombre désiré d'elements, seules les données dont la date est inférieure (donnée plus récente) ou égale à la date spécifiée seront retournées. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur

© 2000-2024, MetaQuotes Ltd.

est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours inférieure ou égale à celle spécifiée.

Lors de la demande de données dans un intervalle spécifié de dates, seules les données de cet intervalle seront retournées. L'intervalle est défini et compté en secondes. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours dans l'intervalle demandé.

Donc, si le jour courant est samedi, au moment d'essayer de copier les données sur l'intervalle de la semaine en spécifiant *start_time=Last_Tuesday* et *stop_time=Last_Friday*, la fonction retournera 0, car l'heure d'ouverture sur l'intervalle de la semaine est le dimanche, mais une barre 1 semaine ne rentre pas dans l'intervalle spécifié.

Si vous devez retourner une valeur correspondant à la barre non complète actuelle, vous pouvez utiliser la 1ère forme d'appel en spécifiant *start_pos*=0 et *count*=1.

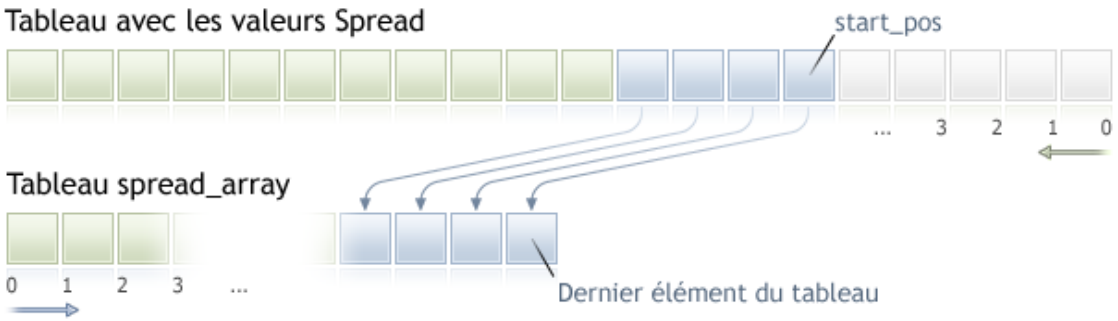
Vous pouvez voir un exemple détaillé de demande des données historiques dans la section Méthodes de Liaison d'Objet. Le script disponible dans cette section montre comment obtenir les valeurs de l'indicateur iFractals sur les 1000 dernières barres et comment afficher les 10 fractales haussières et baissières sur le graphique. Une technique identique peut être utilisée pour tous les indicateurs qui ont des données manquantes et qui sont habituellement dessinés avec les styles suivants :

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopySpread

La fonction copie la quantité spécifiée de données historiques des spreads dans le paramètre spread_array pour la paire symbole-période sélectionnée. Il est à noter que l'ordre des éléments est du présent vers le passé, c'est à dire que la position de départ de 0 signifie la barre courante.



Lors de la copie d'une quantité inconnue de données, il est recommandé d'utiliser un tableau dynamique comme tableau destination, car si le nombre de données est inférieur (ou supérieur) que la longueur du tableau destination, la fonction essaye de réallouer la mémoire pour que l'ensemble des données demandées puisse être contenu.

Si vous connaissez la quantité de données à copier, il est préférable d'utiliser un buffer alloué de façon statique, afin d'éviter une allocation mémoire excessive.

Ceci quelque soit la propriété du tableau destination - as_series=true ou as_series=false. Les données seront copiées afin que le plus vieil élément soit situé au départ de la mémoire physique allouée pour le tableau. Il existe 3 variantes d'appel à la fonction.

Appel avec la première position et le nombre d'éléments désirés

```
int  CopySpread(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   int              start_pos,         // position de départ
   int              count,             // quantité de données à copier
   int              spread_array[]     // tableau destination pour les valeurs de sprea
   );
```

Appel avec la date de début et le nombre d'éléments désirés

```
int  CopySpread(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  timeframe,         // période
   datetime         start_time,        // date et heure de début
   int              count,             // quantité de données à copier
   int              spread_array[]     // tableau destination pour les valeurs de sprea
   );
```

Appel avec les dates de début et de fin d'un intervalle de temps désiré

```
int  CopySpread(
```

© 2000-2024, MetaQuotes Ltd.

```
   string          symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES timeframe,        // période
   datetime        start_time,       // date et heure de début
   datetime        stop_time,        // date et heure de fin
   int             spread_array[]    // tableau destination pour les valeurs de sprea
   );
```

Parameters

symbol_name

[in] Nom du symbole.

timeframe

[in] Période.

start_pos

[in] La position de départ correspondant au 1er élément à copier.

count

[in] Quantité de données à copier.

start_time

[in] Heure de la barre correspondant au premier élément à copier.

stop_time

[in] Heure de la barre correspondant au dernier élément à copier.

spread_array[]

[out] Tableau d'éléments de type int.

Valeur de Retour

Retourne le nombre de données copiées ou -1 en cas d'erreur.

Note

Si l'ensemble de l'intervalle des données demandées est en dehors des données disponibles sur le serveur, la fonction retourne -1. Si des données en dehors de TERMINAL_MAXBARS (nombre maximum de barres sur le graphique) sont demandées, la fonction retournera également -1.

Lors de la demande de données par un indicateur, si les timeseries demandées ne sont pas encore construites ou si elles doivent être téléchargées du serveur, la fonction retournera immédiatement -1, et le téléchargement ou la construction sera initiée.

Lors de la demande de données par un Expert Advisor ou un script, le téléchargement depuis le serveur sera lancé si le terminal n'a pas encore ces données localement, ou la construction des timeseries demandées sera lancée si les données peuvent être construites à partir de l'historique local mais qu'elles ne sont pas encore prêtes. La fonction retournera la quantité de données qui seront prêtes au moment de l'expiration du délai, mais le téléchargement de l'historique continuera, et la fonction retournera plus de données à la prochaine demande similaire.

Lors de la demande de données à partir de la date de départ et du nombre désiré d'elements, seules les données dont la date est inférieure (donnée plus récente) ou égale à la date spécifiée seront retournées. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur

© 2000-2024, MetaQuotes Ltd.

est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours inférieure ou égale à celle spécifiée.

Lors de la demande de données dans un intervalle spécifié de dates, seules les données de cet intervalle seront retournées. L'intervalle est défini et compté en secondes. Cela signifie que l'heure d'ouverture de n'importe quelle barre, pour laquelle une valeur est retournée (volume, spread, valeur du buffer de l'indicateur, prix Open, High, Low, Close ou heure d'ouverture) est toujours dans l'intervalle demandé.

Donc, si le jour courant est samedi, au moment d'essayer de copier les données sur l'intervalle de la semaine en spécifiant *start_time=Last_Tuesday* et *stop_time=Last_Friday*, la fonction retournera 0, car l'heure d'ouverture sur l'intervalle de la semaine est le dimanche, mais une barre 1 semaine ne rentre pas dans l'intervalle spécifié.

Si vous devez retourner une valeur correspondant à la barre non complète actuelle, vous pouvez utiliser la 1ère forme d'appel en spécifiant *start_pos*=0 et *count*=1.

Exemple :

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- dessine le Spread
#property indicator_label1  "Spread"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- paramètres d'entrée
input int      bars=3000;
//--- buffers de l'indicateur
double         SpreadBuffer[];
//+------------------------------------------------------------------+
//| Fonction d'initialisation d'un indicateur personnalisé           |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapping des buffers de l'indicateur
   SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Fonction d'itération de l'indicateur personnalisé                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

```
                  const double &close[],
                  const long &tick_volume[],
                  const long &volume[],
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      int spread_int[];
      ArraySetAsSeries(spread_int,true);
      int spreads=CopySpread(Symbol(),0,0,bars,spread_int);
      Print("Le nombre suivant de valeurs de spreads a été reçu : ",spreads);
      for (int i=0;i<spreads;i++)
      {
      SpreadBuffer[rates_total-1-i]=spread_int[i];
      if(i<=30) Print("spread["+i+"] = ",spread_int[i]);
      }
     }
   else
     {
      double Ask,Bid;
      Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
      Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
      Comment("Ask = ",Ask,"  Bid = ",Bid);
      SpreadBuffer[rates_total-1]=(Ask-Bid)/Point();
     }
//--- retourne la valeur de prev_calculated pour l'appel suivant
   return(rates_total);
  }
```

Vous pouvez voir un exemple détaillé de demande des données historiques dans la section Méthodes de Liaison d'Objet. Le script disponible dans cette section montre comment obtenir les valeurs de l'indicateur iFractals sur les 1000 dernières barres et comment afficher les 10 fractales haussières et baissières sur le graphique. Une technique identique peut être utilisée pour tous les indicateurs qui ont des données manquantes et qui sont habituellement dessinés avec les styles suivants :

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyTicks

La fonction reçoit les ticks au format MqlTick dans ticks_array. Dans ce cas, les ticks sont indexés depuis le passé vers le présent, c'est à dire que le tick l'index 0 est le plus vieux tick dans le tableau. Pour l'analyse des ticks, vérifiez le champ *flags*, qui montre exactement ce qui a changé dans le tick.

```
int  CopyTicks(
   string          symbol_name,          // Nom du symbole
   MqlTick&        ticks_array[],        // Tableau recevant les ticks
   uint            flags=COPY_TICKS_ALL, // Flag déterminant le type des ticks reçus
   ulong           from=0,               // La date à partir de laquelle vous souhai
   uint            count=0               // Le nombre de ticks que vous souhaitez re
   );
```

Paramètres

symbol_name

[in] Symbole.

ticks_array

[out] Un tableau de types MqlTick pour reçevoir les ticks.

flags

[in] Un flag pour définir le type des ticks demandés. COPY_TICKS_INFO - ticks avec les changements Bid et/ou Ask, COPY_TICKS_TRADE - ticks avec les changement de Last et de Volume, COPY_TICKS_ALL - tous les ticks. Pour tout type de demande, les valeurs des ticks précédents sont ajoutées aux champs restants de la structure MqlTick.

from

[in] La date à partir de laquelle vous souhaitez demander les ticks. En millisecondes depuis le 01/01/1970. Si *from*=0, le dernier *count* des ticks sera retourné.

count

[in] Le nombre de ticks demandés. si les paramètres 'from' et 'count' ne sont pas spécifiés, tous les ticks récents disponibles (mais pas plus de 2000) seront écrits dans le tableau ticks_array[].

Valeur de retour

Le nombre de ticks copiés ou -1 en cas d'erreur.

Note Supplémentaire

La fonction CopyTicks() permet de demander et d'analyser tous les ticks reçus. Le premier appel à CopyTicks() initie la synchronisation de la base de données des ticks du symbole stockée sur le disque dur. Si la base de données locale ne fournit pas tous les ticks demandés, alors les ticks manquants seront automatiquement téléchargés depuis le serveur de trading. Les ticks commençant à partir de la date ***from*** spécifiée dans CopyTicks() jusqu'au moment actuel seront synchronisés. Après cela, tous les ticks arrivant pour ce symbole seront ajoutés à la base de donnée des ticks, la gardant ainsi synchronisée.

Si les paramètres *from* et *count* ne sont pas spécifiés, tous les ticks récents disponibles (mais pas plus de 2000) seront écrits dans le tableau *ticks_array[]*. Le paramètre *flags* permet de spécifier le type des ticks demandés.

© 2000-2024, MetaQuotes Ltd.

COPY_TICKS_INFO – seuls les ticks avec des changements de prix Bid et/ou Ask sont retournés. Les données des autres champs seront également ajoutées. Par exemple, si seulement le Bid a changé, les champs *ask* et *volume* seront remplis avec les dernières valeurs connues. Pour savoir exactement ce qui a changé, analysez le champ *flags*, qui contiendra les valeurs TICK_FLAG_BID et/ou TICK_FLAG_ASK. Si un tick a des valeurs des prix Bid et Ask à zéro, et que les flags montrent que ces données ont changé (*flags*=TICK_FLAG_BID|TICK_FLAG_ASK), cela signifie que le livre des ordres (Market Depth) est vide. En d'autres termes, il n'y a aucun ordre buy ou sell.

COPY_TICKS_TRADE – seuls les ticks avec les changements de prix Last et de volume sont retournés. Les données des autres champs seront également ajoutées, c'est à dire que les dernières valeurs connues de Bid et Ask seront spécifiées dans les champs correspondants. Pour savoir exactement ce qui a changé, analysez le champ *flags*, qui contiendra les valeurs TICK_FLAG_LAST et/ou TICK_FLAG_VOLUME.

COPY_TICKS_ALL – tous les ticks ayant un changement sont retournés. Les champs inchangés seront remplis avec les dernières valeurs connues.

L'appel à CopyTicks() avec le flag COPY_TICKS_ALL retourne immédiatement tous les ticks de l'intervalle demandé, tandis qu'un appel dans les autres mode nécessite du temps pour traiter et sélectionner les ticks, ils ne procurent donc pas un avantage significatif en terme de vitesse.

Lors de la demande de ticks (COPY_TICKS_INFO ou COPY_TICKS_TRADE), chaque tick contient toutes les informations de prix, ainsi que l'heure du tick (*bid, ask, last* et *volume*). Cette fonctionnalité est fournie pour une analyse plus facile de l'état du trade au moment de chaque tick, il n'y a donc pas besoin de demander un historique profond des ticks et de rechercher les valeurs des autres champs.

Pour les indicateurs, la fonction CopyTicks() retourne le résultat : lorsqu'elle est appelée depuis un indicateur, CopyTick() retourne immédiatement tous les ticks disponibles pour un symbole, et lancera la synchronisation de la base de données des ticks, si les données disponibles ne sont pas suffisantes. Tous les indicateurs pour un même symbole opèrent dans un thread commun, afin que l'indicateur ne soit pas bloqué par la synchronisation. Après la synchronisation, l'appel suivant à CopyTicks() retournera tous les ticks demandés. Pour les indicateurs, la fonction OnCalculate() est appelée après l'arrivée de chaque tick.

CopyTicks() peut attendre le résultat pendant 45 secondes dans les Expert Advisors et les scripts : puisque distinct des indicateurs, chaque Expert Advisor et chaque script opère dans un thread séparé, et peut donc attendre 45 secondes jusqu'à la fin de la synchronisation. Si le nombre demandé de ticks échoue à être synchronisés pendant ce temps imparti, CopyTicks() retournera les ticks disponibles au moment du timeout et continuera la synchronisation. OnTick() dans les Expert Advisors n'est pas le handler de chaque tick, il ne fait que notifier aux Expert Advisors des changements dans le marché. Ce peut être un ensemble de changements : le terminal peut effectuer simultanément plusieurs ticks, mais OnTick() ne sera appelée qu'une seule fois pour notifier l'EA du dernier état du marché.

Le taux de retour de données : le terminal stocke dans le cache d'accès rapide les 4096 derniers ticks pour chaque instrument (65536 ticks pour les symboles avec un Market Depth en cours d'exécution). Si les ticks demandés pour la session actuelle de trading sont dans le cache, CopyTicks() appelle les ticks stockés dans la mémoire du terminal. Ces demandes demande plus de temps d'exécution. Les demandes les plus lentes sont celles demandant les ticks d'autres jours, puisque les données sont lues depuis le disque dur dans ce cas.

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- Requesting 100 million ticks to be sure we receive the entire tick history
input int      getticks=100000000; // The number of required ticks
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int     attempts=0;     // Count of attempts
   bool    success=false;  // The flag of a successful copying of ticks
   MqlTick tick_array[];   // Tick receiving array
   MqlTick lasttick;       // To receive last tick data
   SymbolInfoTick(_Symbol,lasttick);
//--- Make 3 attempts to receive ticks
   while(attempts<3)
     {
      //--- Measuring start time before receiving the ticks
      uint start=GetTickCount();
//--- Requesting the tick history since 1970.01.01 00:00.001 (parameter from=1 ms)
      int received=CopyTicks(_Symbol,tick_array,COPY_TICKS_ALL,1,getticks);
      if(received!=-1)
        {
         //--- Showing information about the number of ticks and spent time
         PrintFormat("%s: received %d ticks in %d ms",_Symbol,received,GetTickCount()-
         //--- If the tick history is synchronized, the error code is equal to zero
         if(GetLastError()==0)
           {
            success=true;
            break;
           }
         else
            PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d
            _Symbol,received,GetTickCount()-start,_LastError);
        }
      //--- Counting attempts
      attempts++;
      //--- A one-second pause to wait for the end of synchronization of the tick data
      Sleep(1000);
     }
//--- Receiving the requested ticks from the beginning of the tick history failed in t
   if(!success)
     {
      PrintFormat("Error! Failed to receive %d ticks of %s in three attempts",getticks
      return;
     }
```

```
   int ticks=ArraySize(tick_array);
//--- Showing the time of the first tick in the array
   datetime firstticktime=tick_array[ticks-1].time;
   PrintFormat("Last tick time = %s.%03I64u",
               TimeToString(firstticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar
//--- Show the time of the last tick in the array
   datetime lastticktime=tick_array[0].time;
   PrintFormat("First tick time = %s.%03I64u",
               TimeToString(lastticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar

//---
   MqlDateTime today;
   datetime current_time=TimeCurrent();
   TimeToStruct(current_time,today);
   PrintFormat("current_time=%s",TimeToString(current_time));
   today.hour=0;
   today.min=0;
   today.sec=0;
   datetime startday=StructToTime(today);
   datetime endday=startday+24*60*60;
   if((ticks=CopyTicksRange(_Symbol,tick_array,COPY_TICKS_ALL,startday*1000,endday*10
     {
      PrintFormat("CopyTicksRange(%s,tick_array,COPY_TICKS_ALL,%s,%s) failed, error %
                  _Symbol,TimeToString(startday),TimeToString(endday),GetLastError());
      return;
     }
   ticks=MathMax(100,ticks);
//--- Showing the first 100 ticks of the last day
   int counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         counter++;
         PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
        }
     }
//--- Showing the first 100 deals of the last day
   counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         if(((tick_array[i].flags&TICK_FLAG_BUY)==TICK_FLAG_BUY) || ((tick_array[i].fl
           {
            counter++;
            PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
```

© 2000-2024, MetaQuotes Ltd.

```
            }
         }
      }
   }
//+------------------------------------------------------------------+
//| Returns the string description of a tick                         |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
   {
    string desc=StringFormat("%s.%03d ",
                             TimeToString(tick.time),tick.time_msc%1000);
//--- Checking flags
    bool buy_tick=((tick.flags&TICK_FLAG_BUY)==TICK_FLAG_BUY);
    bool sell_tick=((tick.flags&TICK_FLAG_SELL)==TICK_FLAG_SELL);
    bool ask_tick=((tick.flags&TICK_FLAG_ASK)==TICK_FLAG_ASK);
    bool bid_tick=((tick.flags&TICK_FLAG_BID)==TICK_FLAG_BID);
    bool last_tick=((tick.flags&TICK_FLAG_LAST)==TICK_FLAG_LAST);
    bool volume_tick=((tick.flags&TICK_FLAG_VOLUME)==TICK_FLAG_VOLUME);
//--- Checking trading flags in a tick first
    if(buy_tick || sell_tick)
      {
       //--- Forming an output for the trading tick
       desc=desc+(buy_tick?StringFormat("Buy Tick: Last=%G Volume=%d ",tick.last,tick.v
       desc=desc+(sell_tick?StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last,tick
       desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
       desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
       desc=desc+"(Trade tick)";
      }
    else
      {
       //--- Form a different output for an info tick
       desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
       desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
       desc=desc+(last_tick?StringFormat("Last=%G ",tick.last):"");
       desc=desc+(volume_tick?StringFormat("Volume=%d ",tick.volume):"");
       desc=desc+"(Info tick)";
      }
//--- Returning tick description
    return desc;
   }
//+------------------------------------------------------------------+
/* Example of the output
Si-12.16: received 11048387 ticks in 4937 ms
Last tick time = 2016.09.26 18:32:59.775
First tick time = 2015.06.18 09:45:01.000
1.  2016.09.26 09:45.249 Ask=65370 Bid=65370 (Info tick)
2.  2016.09.26 09:47.420 Ask=65370 Bid=65370 (Info tick)
3.  2016.09.26 09:50.893 Ask=65370 Bid=65370 (Info tick)
4.  2016.09.26 09:51.827 Ask=65370 Bid=65370 (Info tick)
```

© 2000-2024, MetaQuotes Ltd.

```
5.  2016.09.26 09:53.810 Ask=65370 Bid=65370 (Info tick)
6.  2016.09.26 09:54.491 Ask=65370 Bid=65370 (Info tick)
7.  2016.09.26 09:55.913 Ask=65370 Bid=65370 (Info tick)
8.  2016.09.26 09:59.350 Ask=65370 Bid=65370 (Info tick)
9.  2016.09.26 09:59.678 Bid=65370 (Info tick)
10. 2016.09.26 10:00.000 Sell Tick: Last=65367 Volume=3 (Trade tick)
11. 2016.09.26 10:00.000 Sell Tick: Last=65335 Volume=45 (Trade tick)
12. 2016.09.26 10:00.000 Sell Tick: Last=65334 Volume=95 (Trade tick)
13. 2016.09.26 10:00.191 Sell Tick: Last=65319 Volume=1 (Trade tick)
14. 2016.09.26 10:00.191 Sell Tick: Last=65317 Volume=1 (Trade tick)
15. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=1 (Trade tick)
16. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=10 (Trade tick)
17. 2016.09.26 10:00.191 Sell Tick: Last=65315 Volume=5 (Trade tick)
18. 2016.09.26 10:00.191 Sell Tick: Last=65313 Volume=3 (Trade tick)
19. 2016.09.26 10:00.191 Sell Tick: Last=65307 Volume=25 (Trade tick)
20. 2016.09.26 10:00.191 Sell Tick: Last=65304 Volume=1 (Trade tick)
21. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=1 (Trade tick)
22. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=10 (Trade tick)
23. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=5 (Trade tick)
24. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=1 (Trade tick)
25. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=6 (Trade tick)
26. 2016.09.26 10:00.191 Sell Tick: Last=65299 Volume=1 (Trade tick)
27. 2016.09.26 10:00.191 Bid=65370 (Info tick)
28. 2016.09.26 10:00.232 Ask=65297 (Info tick)
29. 2016.09.26 10:00.276 Sell Tick: Last=65291 Volume=31 (Trade tick)
30. 2016.09.26 10:00.276 Sell Tick: Last=65290 Volume=1 (Trade tick)
*/
```

Voir aussi

SymbolInfoTick, Structure pour les Prix Courants, OnTick()

© 2000-2024, MetaQuotes Ltd.

CopyTicksRange

La fonction récupère les ticks au format MqlTick dans l'intervalle de dates spécifié et dans le paramètre ticks_array. L'indexation se fait du passé vers le présent, c'est à dire que le tick à l'index 0 est le plus vieux dans le tableau. Pour l'analyse des ticks, vérifiez le champ *flags* qui montre exactement ce qui a changé.

```
int  CopyTicksRange(
   const string     symbol_name,           // nom du symbole
   MqlTick&         ticks_array[],         // tableau destination des ticks
   uint             flags=COPY_TICKS_ALL,  // flag définissant le type des ticks reçus
   ulong            from_msc=0,            // date de départ de l'intervalle
   ulong            to_msc=0               // date de fin de l'intervalle
   );
```

Paramètres

symbol_name

[in] Symbole.

ticks_array

[out] Tableau dynamique ou statique de MqlTick pour reçevoir les ticks. Si le tableau statique ne peut pas contenir tous les ticks de l'intervalle demandé, la quantité maximale possible de ticks est reçue. Dans ce cas, la fonction génère l'erreur ERR_HISTORY_SMALL_BUFFER (4407) .

flags

[in] Un flag définissant le type de ticks demandés. COPY_TICKS_INFO - ticks avec les changements Bid et/ou Ask, COPY_TICKS_TRADE - ticks avec les changements Last et Volume, COPY_TICKS_ALL - tous les ticks. Pour n'importe quelle type de demande, les valeurs des ticks précédents sont ajoutés aux champs restants de la structure MqlTick.

from_msc

[in] La date à partir de laquelle les ticks sont demandés. En millisecondes depuis le 01/01/1970. Si le paramètre *from_msc* n'est pas spécifié, les ticks sont envoyés depuis le début de l'historique. Les ticks ayant une date/heure >= from_msc sont envoyés.

to_msc

[in] La date jusquà laquelle les ticks sont demandés. En millisecondes depuis le 01/01/1970. Les ticks ayant une date/heure <= to_msc sont envoyés. Si le paramètre *to_msc* n'est pas spécifié, tous les ticks jusqu'à la fin de l'historique sont envoyés.

Valeur de Retour

Le nombre de ticks copiés ou -1 en cas d'erreur. GetLastError() peut retourner les erreurs suivants :

- ERR_HISTORY_TIMEOUT - le délai d'attente de synchronisation des ticks est terminé, la fonction a renvoyé tout ce qu'elle avait.
- ERR_HISTORY_SMALL_BUFFER - le buffer statique est trop petit. Seule la quantité d'information possible du tableau a été renvoyée.
- ERR_NOT_ENOUGH_MEMORY - mémire insuffisante pour récupérer l'historique de l'intervalle spécifié dans le tableau dynamique de ticks. Echec d'allocation de la mémoire pour le tableau de ticks.

© 2000-2024, MetaQuotes Ltd.

Note

La fonction CopyTicksRange() est utilisée pour demander les ticks dans un intervalle donné, par exemple pour un jour donné de l'historique. En même temps, CopyTicks() permet de ne spécifier qu'une date de départ, par exemple – recevoir tous les ticks depuis le début du mois jusqu'à maintenant.

Voir aussi

SymbolInfoTick, Structure pour les Prix Actuels, OnTick, CopyTicks

© 2000-2024, MetaQuotes Ltd.

iBars

Retourne le nombre de barres du symbole correspondant et pour la période donnée, disponibles dans l'historique.

```
int  iBars(
   const string          symbol,          // Symbole
   ENUM_TIMEFRAMES       timeframe        // Période
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période courante du graphique.

Valeur de Retour

Le nombre de barres du symbole correspondant et pour la période donnée, disponibles dans l'historique, mais pas plus que le paramètre "Max barres dans le graphique" dans les paramètres de la plateforme.

Exemple :

```
  Print("Le nombre de barres sur 'EURUSD,H1' est ",iBars("EURUSD",PERIOD_H1));
```

Voir également

Barres

© 2000-2024, MetaQuotes Ltd.

iBarShift

Recherche une barre par son heure. La fonction retourne l'indice de la barre correspondant à l'heure spécifiée.

```
int  iBarShift(
   const string        symbol,          // Symbole
   ENUM_TIMEFRAMES     timeframe,       // Période
   datetime            time,            // Heure
   bool                exact=false      // Mode
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. PERIOD_CURRENT signifie la période actuelle du graphique.

time

[in] Valeur de la date/heure à rechercher.

exact=false

[in] Une valeur de retour, dans le cas où aucune barre n'est trouvée pour l'heure spécifiée. Si *exact=false*, *iBarShift* retourne l'indice de la barre la plus proche, dont l'heure d'ouverture est inférieure à l'heure spécifiée (*time_open<time*). Si une telle barre n'est pas trouvée (l'historique n'est pas disponible avant l'heure spécifiée), alors la fonction retourne -1. Si *exact=true*, iBarShift ne recherche pas la barre la plus proche mais retourne immédiatement -1.

Valeur de Retour

L'indice de la barre correspondant à la date/heure spécifiée. Si la barre correspondant à l'heure spécifiée n'est pas trouvée (il y a un trou dans l'historique), la fonction retourne -1 ou l'indice de la barre la plus proche (suivant le paramètre *'exact'*).

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- La date est un dimanche
   datetime time=D'2002.04.25 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- S'il n'y a pas de barre à l'heure spécifiée, iBarShift retournera l'indice de la
   int bar_index=iBarShift(symbol,tf,time,exact);
//--- Vérification du code d'erreur après l'appel à iBarShift()
```

© 2000-2024, MetaQuotes Ltd.

```
   int error=GetLastError();
   if(error!=0)
     {
      PrintFormat("iBarShift(): GetLastError=%d - La date demandée %s "+
                  "pour %s %s n'est pas trouvée dans l'historique disponible",
                  error,TimeToString(time),symbol,EnumToString(tf));
      return;
     }
//--- La fonction iBarShift() a été exécutée avec succès, retourne un résultat pour ex
   PrintFormat("1. %s %s %s(%s): l'indice de la barre est %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),
               DayOfWeek(time),bar_index,string(exact));
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("L'heure de la barre #%d est %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//--- Demande l'indice de la barre à la date spécifiée : s'il n'y a pas de barre, -1 e
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
//--- La fonction iBarShift() a été exécutée avec succès, retourne un résultat pour ex
   PrintFormat("2. %s %s %s (%s):l'indice de la barre est %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time)
               ,DayOfWeek(time),bar_index,string(exact));
  }
//+------------------------------------------------------------------+
//| Retourne le nom du jour de la semaine                            |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(DIMANCHE);
      break;
      case 1: day=EnumToString(LUNDI);
      break;
      case 2: day=EnumToString(MARDI);
      break;
      case 3: day=EnumToString(MERCREDI);
      break;
      case 4: day=EnumToString(JEUDI);
      break;
      case 5: day=EnumToString(VENDREDI);
      break;
      default:day=EnumToString(SAMEDI);
      break;
     }
//---
```

© 2000-2024, MetaQuotes Ltd.

```
   return day;
  }
//+------------------------------------------------------------------+
/*  Résultat de l'exécution
   1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): l'indice de la barre est 64 (exact=fa
   L'heure de la barre #64 est 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY): l'indice de la barre est -1 (exact=t
*/
```

© 2000-2024, MetaQuotes Ltd.

iClose

Retourne le prix Close de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant.

```
double  iClose(
   const string        symbol,          // Symbole
   ENUM_TIMEFRAMES     timeframe,       // Période
   int                 shift            // Décalage
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période actuelle du graphique.

shift

[in] L'indice de la valeur reçue des timeseries (décalage en arrière du nombre spécifié de barres relativement à la barre courante).

Valeur de Retour

Le prix de clôture (Close) de la barre (indiquée avec le paramètre 'shift') sur le graphique correspondant ou 0 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Note

La fonction retourne toujours les données actuelles. Pour cela, elle effectue une demande à la timeserie pour le symbole et la période spécifiés à chaque appel. Cela signifie que s'il n'y a pas de données prêtes au premier appel à la fonction, la préparation des résultats peut prendre un peu de temps.

La fonction ne stocke pas les résultats des appels précédents, et il n'y a pas de cache local pour un retour rapide de la valeur.

Exemple :

```
input int shift=0;
//+------------------------------------------------------------------+
//| Handler de l'évènement "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2024, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Heure : "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Barres: "  ,IntegerToString(bars),"\n"
           );
  }
```

Voir également

CopyClose, CopyRates

© 2000-2024, MetaQuotes Ltd.

iHigh

Retourne le prix High de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant.

```
double  iHigh(
   const string        symbol,          // Symbole
   ENUM_TIMEFRAMES     timeframe,       // Période
   int                 shift            // Décalage
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période actuelle du graphique.

shift

[in] L'indice de la valeur reçue des timeseries (décalage en arrière du nombre spécifié de barres relativement à la barre courante).

Valeur de Retour

Le prix le plus haut (High) de la barre (indiquée avec le paramètre 'shift') sur le graphique correspondant ou 0 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Note

La fonction retourne toujours les données actuelles. Pour cela, elle effectue une demande à la timeserie pour le symbole et la période spécifiés à chaque appel. Cela signifie que s'il n'y a pas de données prêtes au premier appel à la fonction, la préparation des résultats peut prendre un peu de temps.

La fonction ne stocke pas les résultats des appels précédents, et il n'y a pas de cache local pour un retour rapide de la valeur.

Exemple :

```
input int shift=0;
//+------------------------------------------------------------------+
//| Handler de l'évènement "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2024, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Heure : "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Barres: "  ,IntegerToString(bars),"\n"
           );
  }
```

Voir également

CopyHigh, CopyRates

© 2000-2024, MetaQuotes Ltd.

iHighest

Retourne l'indice de la valeur la plus haute trouvée sur le graphique correspondant (décalée relativement à la barre courante).

```
int  iHighest(
   const string        symbol,              // Symbole
   ENUM_TIMEFRAMES     timeframe,           // Période
   ENUM_SERIESMODE     type,                // Identifiant de la timeserie
   int                 count=WHOLE_ARRAY,   // Nombre d'éléments
   int                 start=0              // Indice
  );
```

Parameters

symbol

[in] Le symbole sur lequel la recherche sera effectuée. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période actuelle du graphique.

type

[in] L'identifiant de la timeserie dans laquelle la recherche sera effectuée. Peut être égal à n'importe quelle valeur de ENUM_SERIESMODE.

count=WHOLE_ARRAY

[in] Le nombre d'éléments dans la timeserie (à partir de la barre courante et dans la direction d'augmentation de l'indice) dans lesquelle la recherche doit être effectuée.

start=0

[in] L'indice (décalage relatif à la barre courante) de la barre initiale, à partir de laquelle la recherche de la plus haute valeur commence. Les valeurs négatives sont ignorées et remplacées par zéro.

Valeur de Retour

L'indice de la plus haute valeur trouvée sur le graphique correspondant (décalé relativement à la barre courante) ou -1 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Exemple :

```
   double val;
//--- Calcul de la plus haute valeur Close parmi 20 barres consécutives
//--- De l'indice 4 à l'indice 23 inclus, sur la période courante
   int val_index=iHighest(NULL,0,MODE_CLOSE,20,4);
   if(val_index!=-1)
      val=High[val_index];
   else
      PrintFormat("Erreur d'appel à iHighest(). Code d'erreur=%d",GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

iLow

Retourne le prix Low de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant.

```
double  iLow(
   const string        symbol,          // Symbole
   ENUM_TIMEFRAMES     timeframe,       // Période
   int                 shift            // Décalage
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période actuelle du graphique.

shift

[in] L'indice de la valeur reçue des timeseries (décalage en arrière du nombre spécifié de barres relativement à la barre courante).

Valeur de Retour

Le prix le plus bas (Low) de la barre (indiquée avec le paramètre 'shift') sur le graphique correspondant ou 0 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Note

La fonction retourne toujours les données actuelles. Pour cela, elle effectue une demande à la timeserie pour le symbole et la période spécifiés à chaque appel. Cela signifie que s'il n'y a pas de données prêtes au premier appel à la fonction, la préparation des résultats peut prendre un peu de temps.

La fonction ne stocke pas les résultats des appels précédents, et il n'y a pas de cache local pour un retour rapide de la valeur.

Exemple :

```
input int shift=0;
//+------------------------------------------------------------------+
//| Handler de l'évènement "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2024, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Heure : "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Barres: "  ,IntegerToString(bars),"\n"
           );
  }
```

Voir également

CopyLow, CopyRates

© 2000-2024, MetaQuotes Ltd.

iLowest

Retourne l'indice de la valeur la plus petite trouvée sur le graphique correspondant (décalée relativement à la barre courante).

```
int  iLowest(
   const string        symbol,              // Symbole
   ENUM_TIMEFRAMES     timeframe,           // Période
   ENUM_SERIESMODE     type,                // Identifiant de la timeserie
   int                 count=WHOLE_ARRAY,   // Nombre d'éléments
   int                 start=0              // Indice
  );
```

Parameters

symbol

[in] Le symbole sur lequel la recherche sera effectuée. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période du graphique courant.

type

[in] L'identifiant de la timeserie dans laquelle la recherche sera effectuée. Peut être égal à n'importe quelle valeur de ENUM_SERIESMODE.

count=WHOLE_ARRAY

[in] Le nombre d'éléments dans la timeserie (à partir de la barre courante et dans la direction d'augmentation de l'indice) dans lesquelle la recherche doit être effectuée.

start=0

[in] L'indice (décalé relativement à la barre courante) de la barre initiale, à partir de laquelle la recherche de la plus petite valeur commence. Les valeurs négatives sont ignorées et remplacées par zéro.

Valeur de Retour

L'indice de la plus petite valeur trouvée sur le graphique correspondant (décalé relativement à la barre courante) ou -1 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Exemple :

```
   double val;
//--- Recherche de la barre ayant la plus petite valeur de volume réel parmi 15 barres
//--- De l'indice 10 à l'indice 24 inclus, sur la période courante
   int val_index=iLowest(NULL,0,MODE_REAL_VOLUME,15,10);
   if(val_index!=-1)
      val=Low[val_index];
   else
      PrintFormat("Erreur d'appel à iLowest(). Code d'erreur=%d",GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

iOpen

Retourne le prix Open de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant.

```
double  iOpen(
   const string        symbol,          // Symbole
   ENUM_TIMEFRAMES     timeframe,       // Période
   int                 shift            // Décalage
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période actuelle du graphique.

shift

[in] L'indice de la valeur reçue des timeseries (décalage en arrière du nombre spécifié de barres relativement à la barre courante).

Valeur de Retour

Le prix d'ouverture (Open) de la barre (indiquée avec le paramètre 'shift') sur le graphique correspondant ou 0 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Note

La fonction retourne toujours les données actuelles. Pour cela, elle effectue une demande à la timeserie pour le symbole et la période spécifiés à chaque appel. Cela signifie que s'il n'y a pas de données prêtes au premier appel à la fonction, la préparation des résultats peut prendre un peu de temps.

La fonction ne stocke pas les résultats des appels précédents, et il n'y a pas de cache local pour un retour rapide de la valeur.

Exemple :

```
input int shift=0;
//+------------------------------------------------------------------+
//| Handler de l'évènement "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2024, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Heure : "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Barres: "  ,IntegerToString(bars),"\n"
           );
  }
```

Voir également

CopyOpen, CopyRates

© 2000-2024, MetaQuotes Ltd.

iTime

Retourne l'heure d'ouverture de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant.

```
datetime  iTime(
   const string        symbol,          // Symbole
   ENUM_TIMEFRAMES     timeframe,       // Période
   int                 shift            // Décalage
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période actuelle du graphique.

shift

[in] L'indice de la valeur reçue des timeseries (décalage en arrière du nombre spécifié de barres relativement à la barre courante).

Valeur de Retour

L'heure d'ouverture de la barre (indiquée avec le paramètre 'shift') sur le graphique correspondant ou 0 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Note

La fonction retourne toujours les données actuelles. Pour cela, elle effectue une demande à la timeserie pour le symbole et la période spécifiés à chaque appel. Cela signifie que s'il n'y a pas de données prêtes au premier appel à la fonction, la préparation des résultats peut prendre un peu de temps.

La fonction ne stocke pas les résultats des appels précédents, et il n'y a pas de cache local pour un retour rapide de la valeur.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- La date est un dimanche
   datetime time=D'2018.06.10 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- il n'y a pas de barre à l'heure spécifiée, iBarShift retournera l'indice de la b
   int bar_index=iBarShift(symbol,tf,time,exact);
```

© 2000-2024, MetaQuotes Ltd.

```
   PrintFormat("1. %s %s %s(%s): l'indice de la barre est %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("L'heure de la barre #%d est %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//PrintFormat(iTime(symbol,tf,bar_index));
//--- Demande l'indice de la barre à la date spécifiée : mais il n'y a pas de barre, 
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("2. %s %s %s (%s):l'indice de la barre est %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
  }
//+------------------------------------------------------------------+
//| Retourne le nom du jour de la semaine                            |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(DIMANCHE);
      break;
      case 1: day=EnumToString(LUNDI);
      break;
      case 2: day=EnumToString(MARDI);
      break;
      case 3: day=EnumToString(MERCREDI);
      break;
      case 4: day=EnumToString(JEUDI);
      break;
      case 5: day=EnumToString(VENDREDI);
      break;
      default:day=EnumToString(SAMEDI);
      break;
     }
//---
   return day;
  }
/* Le résultat:
  1. GBPUSD PERIOD_H1 2018.06.10 12:00(DIMANCHE): l'indice de la barre est 64 (exact=
  L'heure de la barre #64 est 2018.06.08 23:00 (VENDREDI)
  2. GBPUSD PERIOD_H1 2018.06.10 12:00 (DIMANCHE):l'indice de la barre est -1 (exact=
*/
```

Voir également

CopyTime, CopyRates

© 2000-2024, MetaQuotes Ltd.

iTickVolume

Retourne le volume des ticks de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant.

```
long  iTickVolume(
   const string        symbol,          // Symbole
   ENUM_TIMEFRAMES     timeframe,       // Période
   int                 shift            // Décalage
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période actuelle du graphique.

shift

[in] L'indice de la valeur reçue des timeseries (décalage en arrière du nombre spécifié de barres relativement à la barre courante).

Valeur de Retour

Le volume des ticks de la barre (indiquée avec le paramètre 'shift') sur le graphique correspondant ou 0 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Note

La fonction retourne toujours les données actuelles. Pour cela, elle effectue une demande à la timeserie pour le symbole et la période spécifiés à chaque appel. Cela signifie que s'il n'y a pas de données prêtes au premier appel à la fonction, la préparation des résultats peut prendre un peu de temps.

La fonction ne stocke pas les résultats des appels précédents, et il n'y a pas de cache local pour un retour rapide de la valeur.

Exemple :

```
input int shift=0;
//+------------------------------------------------------------------+
//| Handler de l'évènement "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2024, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Heure : "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Barres: "  ,IntegerToString(bars),"\n"
           );
  }
```

Voir également

CopyTickVolume, CopyRates

© 2000-2024, MetaQuotes Ltd.

iRealVolume

Retourne le volume réel de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant.

```
long  iRealVolume(
   const string        symbol,          // Symbole
   ENUM_TIMEFRAMES     timeframe,       // Période
   int                 shift            // Décalage
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période actuelle du graphique.

shift

[in] L'indice de la valeur reçue des timeseries (décalage en arrière du nombre spécifié de barres relativement à la barre courante).

Valeur de Retour

Le volume réel de la barre (indiquée avec le paramètre 'shift') sur le graphique correspondant ou 0 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Note

La fonction retourne toujours les données actuelles. Pour cela, elle effectue une demande à la timeserie pour le symbole et la période spécifiés à chaque appel. Cela signifie que s'il n'y a pas de données prêtes au premier appel à la fonction, la préparation des résultats peut prendre un peu de temps.

La fonction ne stocke pas les résultats des appels précédents, et il n'y a pas de cache local pour un retour rapide de la valeur.

Exemple :

```
input int shift=0;
//+------------------------------------------------------------------+
//| Handler de l'évènement "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2024, MetaQuotes Ltd.

```
  Comment(Symbol(),",",EnumToString(Period()),"\n",
          "Heure : "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
          "Open: "  ,DoubleToString(open,Digits()),"\n",
          "High: "  ,DoubleToString(high,Digits()),"\n",
          "Low: "   ,DoubleToString(low,Digits()),"\n",
          "Close: " ,DoubleToString(close,Digits()),"\n",
          "Volume: ",IntegerToString(volume),"\n",
          "Barres: "  ,IntegerToString(bars),"\n"
          );
}
```

Voir également

CopyRealVolume, CopyRates

© 2000-2024, MetaQuotes Ltd.

iVolume

Retourne le volume des ticks de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant.

```
long  iVolume(
   const string        symbol,          // Symbole
   ENUM_TIMEFRAMES     timeframe,       // Période
   int                 shift            // Décalage
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période actuelle du graphique.

shift

[in] L'indice de la valeur reçue des timeseries (décalage en arrière du nombre spécifié de barres relativement à la barre courante).

Valeur de Retour

Le volume des ticks de la barre (indiquée avec le paramètre 'shift') sur le graphique correspondant ou 0 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Note

La fonction retourne toujours les données actuelles. Pour cela, elle effectue une demande à la timeserie pour le symbole et la période spécifiés à chaque appel. Cela signifie que s'il n'y a pas de données prêtes au premier appel à la fonction, la préparation des résultats peut prendre un peu de temps.

La fonction ne stocke pas les résultats des appels précédents, et il n'y a pas de cache local pour un retour rapide de la valeur.

Exemple :

```
input int shift=0;
//+------------------------------------------------------------------+
//| Handler de l'évènement "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

© 2000-2024, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Heure : "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Barres: "  ,IntegerToString(bars),"\n"
           );
  }
```

Voir également

CopyTickVolume, CopyRates

© 2000-2024, MetaQuotes Ltd.

iSpread

Retourne le spread de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant.

```
long  iSpread(
   const string        symbol,          // Symbole
   ENUM_TIMEFRAMES     timeframe,       // Période
   int                 shift            // Décalage
   );
```

Parameters

symbol

[in] Le nom du symbole de l'instrument financier. NULL signifie le symbole courant.

timeframe

[in] Période. Peut être l'une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 signifie la période actuelle du graphique.

shift

[in] L'indice de la valeur reçue des timeseries (décalage en arrière du nombre spécifié de barres relativement à la barre courante).

Valeur de Retour

La valeur du spread de la barre (indiquée avec le paramètre 'shift') sur le graphique correspondant ou 0 en cas d'erreur. Pour connaître les détails de l'erreur, appelez la fonction GetLastError().

Note

La fonction retourne toujours les données actuelles. Pour cela, elle effectue une demande à la timeserie pour le symbole et la période spécifiés à chaque appel. Cela signifie que s'il n'y a pas de données prêtes au premier appel à la fonction, la préparation des résultats peut prendre un peu de temps.

La fonction ne stocke pas les résultats des appels précédents, et il n'y a pas de cache local pour un retour rapide de la valeur.

Exemple :

```
input int shift=0;
//+------------------------------------------------------------------+
//| Handler de l'évènement "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2024, MetaQuotes Ltd.

```
  Comment(Symbol(),",",EnumToString(Period()),"\n",
          "Heure : "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
          "Open: "  ,DoubleToString(open,Digits()),"\n",
          "High: "  ,DoubleToString(high,Digits()),"\n",
          "Low: "   ,DoubleToString(low,Digits()),"\n",
          "Close: " ,DoubleToString(close,Digits()),"\n",
          "Volume: ",IntegerToString(volume),"\n",
          "Barres: "  ,IntegerToString(bars),"\n"
          );
}
```

Voir également

CopySpread, CopyRates

© 2000-2024, MetaQuotes Ltd.

Symboles personnalisés

Fonctions permettant de de créer et de modifier les propriétés d'un symbole personnalisé.

Lors de la connection du terminal à un certain serveur de trades, un utilisateur peut travailler avec des séries temporelles des symboles financiers fournins par un courtier. Les symboles financiers disponibles sont affichés sous forme de liste dans la fenêtre du Market Watch. Un groupe séparé de fonctions permet de récupérer les données des propriétés du symbole, des mises à jour des sessions de trading et du market depth.

Le groupe de fonctions décrites dans cette section permet de créer des symboles personnalisés. Pour cela, les utilisateurs utiliser les symboles existants du serveur de trades, des fichiers texte ou des sources de données externes.

Fonction	Action
CustomSymbolCreate	Crée un symbole personnalisé avec le nom spécifié dans le groupe spécifié
CustomSymbolDelete	Supprime un symbole personnalisé ayant le nom spécifié
CustomSymbolSetInteger	Assigne une valeur de type integer à la propriété d'un symbole personnalisé
CustomSymbolSetDouble	Assigne une valeur de type réel à la propriété d'un symbole personnalisé
CustomSymbolSetString	Assigne une valeur de type string à la propriété d'un symbole personnalisé
CustomSymbolSetMarginRate	Définit les taux de marge suivant le type et la direction de l'ordre pour un symbole personnalisé
CustomSymbolSetSessionQuote	Définit les heures de début et de fin de la session de cotation spécifiée pour le symbole et le jour de la semaine spécifiés
CustomSymbolSetSessionTrade	Définit les heures de début et de fin de la session de cotation spécifiée pour le symbole et le jour de la semaine spécifiés
CustomRatesDelete	Supprime toutes les barres de l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié
CustomRatesReplace	Remplace tout l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié avec les données du tableau de données de type MqlRates
CustomRatesUpdate	Ajoute des barres manquantes dans l'historique du symbole personnalisé et remplace les données existantes avec celles du tableau de données de type MqlRates
CustomTicksAdd	Adds data from an array of the MqlTick type to the price history of a custom symbol. The custom symbol must be selected in the Market Watch window
CustomTicksDelete	Supprime tous les ticks de l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
CustomTicksReplace	Remplace tout l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié avec les données du tableau de données de type MqlTick
CustomBookAdd	Passe le statut du Depth of Market pour un symbole personnalisé

© 2000-2024, MetaQuotes Ltd.

CustomSymbolCreate

Crée un symbole personnalisé avec le nom spécifié dans le groupe spécifié.

```
bool  CustomSymbolCreate(
   const string     symbol_name,          // nom du symbole personnalisé
   const string     symbol_path="",       // nom du groupe dans lequel le symbole doit
   const string     symbol_origin=NULL    // nom d'un symbole servant de base pour crée
   );
```

Parameters

symbol_name

[in] Nom du symbole personnalisé. Il ne doit pas contenir les groupes ou sous-groupes dans lesquels le symbole est situé.

symbol_path=""

[in] Le nom du groupe dans lequel se trouve le symbole.

symbol_origin=NULL

[in] Nom d'un symbole à partir duquel les propriétés du symbole personnalisé créé seront copiées. Après avoir créé un symbole personnalisé, n'importe quelle valeur de propriété peut être changée en utilisant les fonctions correspondantes.

Valeur de Retour

true - en cas de succès, sinon - false. Pour obtenir des informations sur l'erreur, appelez la fonction GetLastError().

Note

Tous les symboles personnalisés sont créés dans une section spéciale nommée Custom. Si aucun nom de groupe n'est spécifié (le paramètre *symbol_path* de la fonction CustomSymbolCreate contient une chaîne vide ou NULL), le symbole personnalisé est généré à la racine de la section Custom. Nous pouvons dessiner une analogie avec le système de fichiers, où les groupes et les sous-groupes peuvent être vus comme des dossiers et des sous-dossiers

Les noms du symbole et du groupe ne peuvent contenir que des caractères latins avec la ponctuation, des espaces ou des caractères spéciaux (ne peut contenir que ".", "_", "&" et "#"). Il n'est pas recommandé d'utiliser les caractères <, >, :, ", /, |, ?, *.

Le nom du symbole personnalisé doit être unique dans le groupe dans lequel il est créé. Si un symbole existe déjà avec le même nom, la fonction CustomSymbolCreate() retourne 'false', et l'appel suivant à GetLastError() retourne l'erreur 5300 (ERR_NOT_CUSTOM_SYMBOL) ou 5304 (ERR_CUSTOM_SYMBOL_EXIST).

La longueur du nom du symbole ne doit pas être supérieure à 31 caractères. Sinon, CustomSymbolCreate() retourne 'false' et l'erreur 5302 - ERR_CUSTOM_SYMBOL_NAME_LONG est générée.

Le paramètre *symbol_path* peut être défini de 2 façons :

- seul un nom de groupe sans un nom pour le symbole personnalisé, par exemple - "CFD\\Metals". C'est la meilleure façon de faire pour éviter les erreurs.

© 2000-2024, MetaQuotes Ltd.

- ou nom du <groupe> + séparateur de groupes "\\"+<nom du symbole personnalisé>, par exemple - "CFD\\Metals\\Platinum". Dans ce cas, le nom du groupe doit se terminer avec le nom exact du symbole personnalisé. En cas de différence, le symbole personnalisé sera quand même créé, mais pas dans le groupe désiré. Par exemple, si *symbol_path*="CFD\\Metals**P**latinum" et *symbol_name*="**p**latinum" (erreur d'enregistrement), alors un symbole personnalisé nommé "platinum" est créé dans le groupe "Custom\CFD\Metals\Platinum". La fonction SymbolInfoGetString("platinum",SYMBOL_PATH) retourne la valeur "Custom\CFD\Metals\Platinum\platinum".

Notez que la propriété SYMBOL_PATH retourne le chemin avec le nom du symbole à la fin. Il ne peut donc pas être copié sans changement si vous voulez créer un symbole personnalisé dans le même groupe. Dans ce cas, il est nécessaire de couper le nom du symbole pour ne pas obtenir le résultat décrit ci-dessus.

Si un symbole non existant est utilisé comme paramètre *symbol_origin*, alors le symbole personnalisé est créé vide comme si le paramètre *symbol_origin* n'était pas donné. L'erreur 4301 - ERR_MARKET_UNKNOWN_SYMBOL est générée dans ce cas.

La longueur du paramètre *symbol_path* ne doit pas être supérieure à 127 caractères en tenant compte de "Custom\\", des séparateurs de groupes "\\" et du nom du symbole s'il est spécifié à la fin.

Voir également

SymbolName, SymbolSelect, CustomSymbolDelete

© 2000-2024, MetaQuotes Ltd.

CustomSymbolDelete

Supprime le symbole personnalisé ayant le nom spécifié.

```
bool  CustomSymbolDelete(
   const string     symbol_name          // Nom du symbole personnalisé
   );
```

Paramètres

symbol

[in] Nom du symbole personnalisé. Il ne doit pas correspondre au nom d'un symbole existant.

Valeur de Retour

true - en cas de succès - false sinon. Pour obtenir des informations sur l'erreur, il faut appeler la fonction GetLastError().

Note

Le symbole personnalisé affiché dans le Market Watch ou dans un graphique ne peut pas être supprimé.

Voir aussi

SymbolName, SymbolSelect, CustomSymbolCreate

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetInteger

Assigne une valeur de type integer à la propriété d'un symbole personnalisé.

```
bool  CustomSymbolSetInteger(
   const string               symbol_name,       // nom du symbole
   ENUM_SYMBOL_INFO_INTEGER   property_id,       // identifiant de la propriété
   long                       property_value     // valeur de la propriété
   );
```

Paramètres

symbol_name

[in] Nom du symbole personnalisé.

property_id

[in] Identifiant de la propriété du symbole. La valeur peut être l'une des valeurs de l'énumération ENUM_SYMBOL_INFO_INTEGER.

property_value

[in] Une variable de type long contenant la valeur de la propriété.

Valeur de Retour

true - en cas de succès - false sinon. Pour obtenir des informations sur l'erreur, il faut appeler la fonction GetLastError().

Note

The minute and tick history of the custom symbol is completely removed if any of these properties is changed in the symbol specification:

- SYMBOL_CHART_MODE - price type for constructing bars (Bid or Last)
- SYMBOL_DIGITS - number of digits after the decimal point to display the price

After deleting the custom symbol history, the terminal attempts to create a new history using the updated properties. The same happens when the custom symbol properties are changed manually.

Voir aussi

SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetDouble

Assigne une valeur de type réel à la propriété d'un symbole personnalisé.

```
bool  CustomSymbolSetDouble(
   const string              symbol_name,       // nom du symbole
   ENUM_SYMBOL_INFO_DOUBLE   property_id,       // identifiant de la propriété
   double                    property_value     // valeur de la propriété
   );
```

Paramètres

symbol_name

[in] Nom du symbole personnalisé.

property_id

[in] Identifiant de la propriété du symbole. La valeur peut être l'une des valeurs de l'énumération ENUM_SYMBOL_INFO_DOUBLE.

property_value

[in] Une variable de type double contenant la valeur de la propriété.

Valeur de Retour

true - en cas de succès - false sinon. Pour obtenir des informations sur l'erreur, il faut appeler la fonction GetLastError().

Note

The minute and tick history of the custom symbol is completely removed if any of these properties is changed in the symbol specification:

- SYMBOL_POINT - one point value
- SYMBOL_TRADE_TICK_SIZE - value of a tick that specifies the minimum allowable price change
- SYMBOL_TRADE_TICK_VALUE - one-tick price change value for a profitable position

After deleting the custom symbol history, the terminal attempts to create a new history using the updated properties. The same happens when the custom symbol properties are changed manually.

Voir aussi

SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetString

Assigne une valeur de type string à la propriété d'un symbole personnalisé.

```
bool  CustomSymbolSetString(
   const string              symbol_name,       // nom du symbole
   ENUM_SYMBOL_INFO_STRING   property_id,       // identifiant de la propriété
   string                    property_value     // valeur de la propriété
   );
```

Paramètres

symbol_name

[in] Nom du symbole personnalisé.

property_id

[in] Identifiant de la propriété du symbole. La valeur peut être l'une des valeurs de l'énumération ENUM_SYMBOL_INFO_STRING.

property_value

[in] Une variable de type string contenant la valeur de la propriété.

Valeur de Retour

true - en cas de succès - false sinon. Pour obtenir des informations sur l'erreur, il faut appeler la fonction GetLastError().

Note

The minute and tick history of the custom symbol is completely removed if the SYMBOL_FORMULA property (setting the equation for the custom symbol price construction) is changed in the symbol specification. After deleting the custom symbol history, the terminal attempts to create a new history using the new equation. The same happens when the custom symbol equation is changed manually.

Voir aussi

SymbolInfoString

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetMarginRate

Définit les taux de marge suivant le type et la direction de l'ordre pour un symbole personnalisé.

```
bool  CustomSymbolSetMarginRate(
   const string       symbol_name,              // nom du symbole
   ENUM_ORDER_TYPE    order_type,               // type de l'ordre
   double             initial_margin_rate,      // taux de marge initiale
   double             maintenance_margin_rate   // taux de marge de maintien
   );
```

Paramètres

symbol_name

[in] Nom du symbole personnalisé.

order_type

[in] Type de l'ordre.

initial_margin_rate

[in] Une variable de type double avec le taux de marge initiale. La marge initiale est un dépôt de sécurité pour une transaction de 1 lot dans la direction appropriée. Le montant des fonds devant être réservés sur le compte lors du placement d'un ordre du type spécifié est obtenu en multipliant le taux par la marge initiale.

maintenance_margin_rate

[in] Une variable de type double avec le taux de marge de maintien. La marge de maintien est le montant minimum pour conserver une position de 1 lot ouverte dans la direction désirée. Le montant des fonds devant être réservés sur le compte après l'activation d'un ordre du type spécifié est obtenu en multipliant le taux par la marge de maintien.

Valeur de Retour

true - en cas de succès - false sinon. Pour obtenir des informations sur l'erreur, il faut appeler la fonction GetLastError().

Voir aussi

SymbolInfoMarginRate

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetSessionQuote

Définit les heures de début et de fin de la session de cotation spécifiée pour le symbole et le jour de la semaine spécifiés.

```
bool  CustomSymbolSetSessionQuote(
   const string      symbol_name,          // nom du symbole
   ENUM_DAY_OF_WEEK  day_of_week,          // jour de la semaine
   uint              session_index,        // indice de la session
   datetime          from,                 // date/heure de début de la session
   datetime          to                    // date/heure de fin de la session
   );
```

Paramètres

symbol_name

[in] Nom du symbole personnalisé.

ENUM_DAY_OF_WEEK

[in] Jour de la semaine, une valeur de l'énumération ENUM_DAY_OF_WEEK.

uint

[in] Indice de la session pour laquelle les heures de début et de fin vont être définies. L'indexation des sessions commence à 0.

from

[in] Heure de début de la session en secondes depuis 00:00, la valeur des données de la variable est ignorée.

to

[in] Heure de fin de la session en secondes depuis 00:00, la valeur des données de la variable est ignorée.

Valeur de Retour

true - en cas de succès - false sinon. Pour obtenir des informations sur l'erreur, il faut appeler la fonction GetLastError().

Note

Si la session avec le *session_index* spécifié existe déjà, la fonction modifie simplement le début et la fin de la session.

Si aucun paramètre de début et de fin n'ont été passés pour la session (*from=0* et *to=0*), la session correspondante au *session_index* est supprimée, tandis que l'indexation des sessions est décalée vers le bas.

Les sessions ne peuvent être ajoutées que de façon séquentielle. En d'autres termes, vous ne pouvez ajouter avec *session_index=1* que si une session avec l'indice 0 existe déjà. Si cette règle est cassée, aucune nouvelle session n'est créée, et la fonction elle-même reourne 'false'.

Voir aussi

SymbolInfoSessionQuote, Symbol info, TimeToStruct, structure Date

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetSessionTrade

Définit les heures de début et de fin de la session de trading spécifiée pour le symbole et le jour de la semaine spécifiés.

```
bool  CustomSymbolSetSessionTrade(
   const string      symbol_name,          // nom du symbole
   ENUM_DAY_OF_WEEK  day_of_week,          // jour de la semaine
   uint              session_index,        // indice de la session
   datetime          from,                 // date/heure de début de la session
   datetime          to                    // date/heure de fin de la session
   );
```

Paramètres

symbol_name

[in] Nom du symbole personnalisé.

ENUM_DAY_OF_WEEK

[in] Jour de la semaine, une valeur de l'énumération ENUM_DAY_OF_WEEK.

uint

[in] Indice de la session pour laquelle les heures de début et de fin vont être définies. L'indexation des sessions commence à 0.

from

[in] Heure de début de la session en secondes depuis 00:00, la valeur des données de la variable est ignorée.

to

[in] Heure de fin de la session en secondes depuis 00:00, la valeur des données de la variable est ignorée.

Valeur de Retour

true - en cas de succès - false sinon. Pour obtenir des informations sur l'erreur, il faut appeler la fonction GetLastError().

Note

Si la session avec le *session_index* spécifié existe déjà, la fonction modifie simplement le début et la fin de la session.

Si aucun paramètre de début et de fin n'ont été passés pour la session (*from=0* et *to=0*), la session correspondante au *session_index* est supprimée, tandis que l'indexation des sessions est décalée vers le bas.

Les sessions ne peuvent être ajoutées que de façon séquentielle. En d'autres termes, vous ne pouvez ajouter avec *session_index=1* que si une session avec l'indice 0 existe déjà. Si cette règle est cassée, aucune nouvelle session n'est créée, et la fonction elle-même reourne 'false'.

Voir aussi

SymbolInfoSessionTrade, Symbol info, TimeToStruct, structure Date

© 2000-2024, MetaQuotes Ltd.

CustomRatesDelete

Supprime toutes les barres de l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié.

```
int  CustomRatesDelete(
   const string     symbol,        // nom du symbole
   datetime         from,          // date de début
   datetime         to             // date de fin
   );
```

Paramètres

symbol

[in] Nom du symbole personnalisé.

from

[in] Date/heure de la première barre à supprimer de l'historique des prix dans l'intervalle spécifié.

to

[in] Date/heure de la dernière barre à supprimer de l'historique des prix dans l'intervalle spécifié.

Valeur de Retour

Nombre de barres supprimées ou -1 en cas d'erreur.

Voir aussi

CustomRatesReplace, CustomRatesUpdate, CopyRates

© 2000-2024, MetaQuotes Ltd.

CustomRatesReplace

Remplace toutes les barres de l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié avec les données du tableau de données de type MqlRates.

```
int  CustomRatesReplace(
   const string     symbol,             // nom du symbole
   datetime         from,               // date de début
   datetime         to,                 // date de fin
   const MqlRates&  rates[],            // tableau pour les données à appliquer à un s
   uint             count=WHOLE_ARRAY   // nombre d'éléments du tableau rates[] à util
   );
```

Paramètres

symbol

[in] Nom du symbole personnalisé.

from

[in] Date/heure de la première barre à mettre à jour de l'historique des prix dans l'intervalle spécifié.

to

[in] Date/heure de la dernière barre à mettre à jour de l'historique des prix dans l'intervalle spécifié.

rates[]

[in] Tableau de données d'historique de type MqlRates pour M1.

count=WHOLE_ARRAY

[in] Nombre d'éléments du tableau *rates[]* à utiliser pour le remplacement. WHOLE_ARRAY signifie que tous les éléments du tableau *rates[]* doivent être utilisés pour le remplacement.

Valeur de Retour

Nombre de barres mises à jour ou -1 en cas d'erreur.

Note

Si une barre du tableau *rates[]* va au-delà de l'intervalle spécifié, elle est ignorée. Si une barre est déjà présente dans l'historique des prix et entre dans l'intervalle donné, elle est remplacée. Toutes les autres barres de l'historique des prix actuel situées en dehors de l'intervalle spécifié restent inchangées. Le tableau de données *rates[]* doit être correct au regard des prix OHLC, et les heures d'ouverture des barres doivent correspondre à la période M1.

Voir aussi

CustomRatesDelete, CustomRatesUpdate, CopyRates

© 2000-2024, MetaQuotes Ltd.

CustomRatesUpdate

Ajoute des barres manquantes dans l'historique du symbole personnalisé et remplace les données existantes avec celles du tableau de données de type MqlRates.

```
int  CustomRatesUpdate(
   const string     symbol,             // Nom du symbole personnalisé
   const MqlRates&  rates[],            // tableau pour les données à appliquer à un s
   uint             count=WHOLE_ARRAY   // nombre d'éléments du tableau rates[] à util
   );
```

Paramètres

symbol

[in] Nom du symbole personnalisé.

rates[]

[in] Tableau de données d'historique de type MqlRates pour M1.

count=WHOLE_ARRAY

[in] Nombre d'éléments du tableau *rates[]* à utiliser pour la mise à jour. WHOLE_ARRAY signifie que tous les éléments du tableau *rates[]* doivent être utilisés pour la mise à jour.

Valeur de Retour

Nombre de barres mises à jour ou -1 en cas d'erreur.

Note

S'il n'y a aucune barre du tableau *rates[]* dans l'historique du symbole personnalisé actuel, il est ajouté. Si une barre existe déjà, elle est remplacée. Toutes les autres barres de l'historique des prix actuel restent inchangées. Le tableau de données *rates[]* doit être correct au regard des prix OHLC, et les heures d'ouverture des barres doivent correspondre à la période M1.

Voir aussi

CustomRatesReplace, CustomRatesDelete, CopyRates

© 2000-2024, MetaQuotes Ltd.

CustomTicksAdd

Ajoute les données d'un tableau de données de type MqlTick à l'historique des prix d'un symbole personnalisé. Le symbole personnalisé doit être sélectionné dans la fenêtre du Market Watch.

```
int  CustomTicksAdd(
   const string     symbol,             // Nom du symbole
   const MqlTick&   ticks[],            // Le tableau de ticks à appliquer au symbole
   uint             count=WHOLE_ARRAY   // nombre d'éléments du tableau ticks[] à util
   );
```

Paramètres

symbol

[in] Le nom du symbole personnalisé.

ticks[]

[in] Un tableau de ticks du type MqlTick ordonnés par date, du plus récent au plus ancien, c'est à dire ticks[k].time_msc <= ticks[n].time_msc, avec k<n.

count=WHOLE_ARRAY

[in] Nombre d'éléments du tableau *ticks[]* à utiliser pour l'ajout. WHOLE_ARRAY signifie que tous les éléments du tableau *ticks[]* doivent être utilisés.

Valeur de Retour

Le nombre de ticks ajoutés ou -1 en cas d'erreur.

Note

La fonction CustomTicksAdd ne fonctionne que pour les symboles personnalisés ouverts dans la fenêtre du Market Watch. Si le symbole n'est pas sélectionné dans le Market Watch, vous devez ajouter les ticks avec CustomTicksReplace.

La fonction CustomTicksAdd permet de transmettre des ticks comme s'ils avaient été émis par le serveur du courtier. Les données sont envoyées dans la fenêtre du Market Watch au lieu d'être écrites directement dans la base de données des ticks. Le terminal sauvegarde ensuite les ticks depuis le Market Watch dans une base de données. Si trop de données sont transmises à un appel de la fonction, son comportement est modifié pour réduire l'utilisation des ressources. Si plus de 256 ticks sont passés, les données sont divisées en deux parties. La première, c'est à dire la partie la plus grande, est écrite directement dans la base de données des ticks (comme cela est fait dans CustomTicksReplace). La deuxième partie, contenant 128 ticks, est passée à la fenêtre du Market Watch, à partir de laquelle le terminal sauvegarde les ticks dans une base de données.

La structure MqlTick a deux champs temporels : time (l'heure du tick en secondes) et time_msc (l'heure du tick en millisecondes), qui sont comptées depuis le 1er janvier 1970. Ces champs des ticks ajoutés sont traités dans l'ordre suivant :

1. Si ticks[k].time_msc!=0, le champ est utilisé pour remplir ticks[k].time field, c'est à dire que ticks[k].time=ticks[k].time_msc/1000 (division entière) est défini pour le tick
2. Si ticks[k].time_msc==0 et ticks[k].time!=0, l'heure en millisecondes est obtenue en multipliant par 1000, c'est à dire que ticks[k].time_msc=ticks[k].time*1000
3. Si ticks[k].time_msc==0 et ticks[k].time==0, l'heure atuelle du serveur de trades plus 1 milliseconde au moment de l'appel à CustomTicksAdd est écrit dans ces champs.

© 2000-2024, MetaQuotes Ltd.

Si la valeur de ticks[k].bid, ticks[k].ask, ticks[k].last ou ticks[k].volume est supérieure à zéro, la combinaison des flags correspondants est écrite dans le champ ticks[k].flags :

- TICK_FLAG_BID – le tick a changé le prix bid
- TICK_FLAG_ASK – le tick a changé le prix ask
- TICK_FLAG_LAST – le tick a changé le prix de la dernière transaction
- TICK_FLAG_VOLUME – le tick a changé le volume

Si la valeur d'un champ est inférieure ou égale à zéro, le flag correspondant n'est pas écrit dans le champ ticks[k].flags.

Les flags TICK_FLAG_BUY et TICK_FLAG_SELL ne sont pas ajoutés à l'historique d'un symbole personnalisé.

Voir aussi

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

© 2000-2024, MetaQuotes Ltd.

CustomTicksDelete

Supprime tous les ticks de l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié.

```
int  CustomTicksDelete(
   const string     symbol,            // nom du symbole
   long             from_msc,          // date de début
   long             to_msc             // date de fin
   );
```

Paramètres

symbol

[in] Nom du symbole personnalisé.

from_msc

[in] Date/heure du premier tick à supprimer de l'historique des prix dans l'intervalle spécifié. Date/heure en millisecondes depuis le 01.01.1970.

to_msc

[in] Date/heure du dernier tick à supprimer de l'historique des prix dans l'intervalle spécifié. Date/heure en millisecondes depuis le 01.01.1970.

Valeur de Retour

Nombre de ticks supprimés ou -1 en cas d'erreur.

Voir aussi

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

© 2000-2024, MetaQuotes Ltd.

CustomTicksReplace

Remplace toutes les prix de l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié avec les données du tableau de données de type MqlTick..

```
int  CustomTicksReplace(
   const string     symbol,            // nom du symbole
   long             from_msc,          // date de début
   long             to_msc,            // date de fin
   const MqlTick&   ticks[],           // tableau pour les données à appliquer à un sy
   uint             count=WHOLE_ARRAY   // nombre d'éléments du tableau ticks[] à util
   );
```

Paramètres

symbol

[in] Nom du symbole personnalisé.

from_msc

[in] Date/heure du premier tick à supprimer de l'historique des prix dans l'intervalle spécifié. Date/heure en millisecondes depuis le 01.01.1970.

to_msc

[in] Date/heure du dernier tick à supprimer de l'historique des prix dans l'intervalle spécifié. Date/heure en millisecondes depuis le 01.01.1970.

ticks[]

[in] Tableau de données d'historique des ticks de type MqlTick ordonnés par heure en ordre ascendant.

count=WHOLE_ARRAY

[in] Nombre d'éléments du tableau *ticks[]* à utiliser pour le remplacement dans l'intervalle de temps spécifié. WHOLE_ARRAY signifie que tous les éléments du tableau *ticks[]* doivent être utilisés.

Valeur de Retour

Nombre de ticks mis à jour ou -1 en cas d'erreur.

Note

Puisque plusieurs ticks peuvent avoir la même heure jusqu'à la milliseconde dans un flux de cotations (l'heure précise du tick est stockée dans le champ *time_msc* de la structure MqlTick), la fonction CustomTicksReplace ne trie pas automatiquement les éléments du tableau *ticks[]* par heure. Le tableau des ticks doit donc être pré-ordonné par heure en ordre ascendant.

Les ticks sont remplacés de façon consécutive, jour après jour, jusqu'à ce que l'heure spécifiée *to_msc* ou jusqu'à ce qu'une erreur survienne. Le premier jour de l'intervalle spécifié est traité, ensuite le suivant, etc. Dès qu'une différence entre l'heure du tick et l'ordre ascendant (non descendant) est détectée, le remplacement du tick s'arrête sur le jour courant. Tous les ticks des jours précédents sont remplacés avec succès, tandis que ceux du jour courant (au moment du tick invalide) et ceux de tous les jours restants dans l'intervalle spécifié restent inchangés.

© 2000-2024, MetaQuotes Ltd.

Si le tableau *ticks[]* ne contient aucune donnée pour un jour (plus généralement, n'importe quel intervalle de temps), un "trou" correspondant aux données manquantes apparaîtra dans l'historique du symbole personnalisé après que les données des ticks de `ticks[]` ont été appliquées. En d'autres termes, l'appel de CustomTicksReplace avec les ticks manquants est équivalent à supprimer une partie de l'historique des ticks, comme si CustomTicksDelete avec l'intervalle du "trou" était appelé.

Si la base de données des ticks n'a pas de données pour l'intervalle de temps spécifié, CustomTicksReplace ajoutera le tableau ticks[] à la base de données des ticks.

La fonction CustomTicksReplace utilise directement la base de données des ticks.

Voir aussi

CustomRatesDelete, CustomRatesUpdate, CustomTicksDelete, CopyTicks, CopyTicksRange

© 2000-2024, MetaQuotes Ltd.

CustomBookAdd

Passe le statut du Depth of Market pour un symbole personnalisé. Le fonction permet de diffuser le Depth of Market comme si les prix arrivaient du serveur du courtier.

```
bool  CustomBookAdd(
   const string        symbol,            // nom du symbole
   const MqlBookInfo&  books[]            // tableau contenant les descriptions des é
   uint                count=WHOLE_ARRAY  // nombre d'éléments à utiliser
   );
```

Paramètres

symbol

[in] Nom du symbole personnalisé.

books[]

[in] Le tableau de données de type MqlBookInfo décrivant totalement le statut du Depth of Market — toutes les demandes d'achats et de ventes. Le statut passé du Depth of Market remplace totalement le précédent.

count=WHOLE_ARRAY

[in] Le nombre d'éléments du tableau 'books' à passer à la fonction. Le tableau entier est utilisé par défaut.

Valeur de Retour

true - en cas de succès, sinon - false. Pour obtenir des informations sur l'erreur, appelez la fonction GetLastError().

Note

La fonction CustomBookAdd ne fonctionne que pour les symboles personnalisés pour lesquels le Depth of Market est ouvert — via la plateforme ou via la fonction MarketBookAdd.

Lorsque vous injectez le Depth of Market, les prix Bid et Ask du symbole ne sont pas mis à jour. Vous devez contrôler le changement des meilleurs prix et injecter les ticks avec la fonction CustomTicksAdd.

La fonction vérifie l'exactitude des données transmises : le type, le prix et le volume doivent être indiqués pour chaque élément. De plus, MqlBookInfo.volume et MqlBookInfo.volume_real ne doivent pas être nuls ou négatifs ; si les deux volumes sont négatifs, cela sera considéré comme une erreur. Vous pouvez spécifier n'importe lequel des volumes ou les deux : celui qui est indiqué ou qui est positif sera utilisé :

```
volume=-1 && volume_real=2 — volume_real=2 will sera utilisé,
volume=3 && volume_real=0 — volume=3 will sera utilisé.
```

L'ordre des éléments MqlBookInfo dans le tableau 'books' est sans importance. Lors de la sauvegarde des données, le terminal les trie lui-même par prix.

Lors de la sauvegarde des données, le paramètre "Profondeur du livre (Book depth)" (SYMBOL_TICKS_BOOKDEPTH) du symbole personnalisé destination est vérifié. Si le nombre

© 2000-2024, MetaQuotes Ltd.

de demandes de ventes (sell) est supérieur à cette valeur dans le Depth of Market passé, les niveaux en excès sont rejetés. Ceci est également valable pour les demandes d'achat (buy).

Exemple de remplissage du tableau 'books' :

Statut du Depth of Market	Remplissage de books[]

Volume / Price
100.00 / 1.14337
50.00 / 1.14336
40.00 / 1.14335
10.00 / 1.14333
10.00 / 1.14322
90.00 / 1.14320
100.00 / 1.14319
10.00 / 1.14318
	books[0].type=BOOK_TYPE_SELL;
	books[0].price=1.14337;
	books[0].volume=100;
	books[1].type=BOOK_TYPE_SELL;
	books[1].price=1.14330;
	books[1].volume=50;
	books[2].type=BOOK_TYPE_SELL;
	books[2].price=1.14335;
	books[2].volume=40;
	books[3].type=BOOK_TYPE_SELL;
	books[3].price=1.14333;
	books[3].volume=10;
	books[4].type=BOOK_TYPE_BUY;
	books[4].price=1.14322;
	books[4].volume=10;
	books[5].type=BOOK_TYPE_BUY;
	books[5].price=1.14320;
	books[5].volume=90;
	books[6].type=BOOK_TYPE_BUY;
	books[6].price=1.14319;
	books[6].volume=100;
	books[7].type=BOOK_TYPE_BUY;
	books[7].price=1.14318;
	books[7].volume=10;

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- active le Depth of Market pour un symbole pour lequel nous allons récupérer les
   MarketBookAdd(Symbol());
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//| Fonction Tick                                                    |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   MqlTick ticks[];
   ArrayResize(ticks,1);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- copie les prix courants du symbole en cours dans le symbole personnalisé
   if(SymbolInfoTick(Symbol(),ticks[0]))
     {
      string symbol_name=Symbol()+".SYN";
      CustomTicksAdd(symbol_name,ticks);
     }
  }
//+------------------------------------------------------------------+
//| Fonction Book                                                    |
//+------------------------------------------------------------------+
void OnBookEvent(const string &book_symbol)
  {
//--- copie le statut courant du Depth of Market du symbole courant vers le symbole pe
   if(book_symbol==Symbol())
     {
      MqlBookInfo book_array[];
      if(MarketBookGet(Symbol(),book_array))
        {
         string symbol_name=Symbol()+".SYN";
         CustomBookAdd(symbol_name,book_array);
        }
     }
  }
//+------------------------------------------------------------------+
```

Voir aussi

MarketBookAdd, CustomTicksAdd, OnBookEvent

© 2000-2024, MetaQuotes Ltd.

Les opérations avec les graphiques

Les fonctions définissant les propriétés du graphique (ChartSetInteger, ChartSetDouble, ChartSetString) sont asynchrone et sont utilisées pour envoyer des commandes de mise à jour au graphique. Si ces fonctions sont exécutées avec succès, la commande est ajoutée dans la queue commune des évènements du graphique. Les changements de propriété du graphique sont implémentées avec la prise en charge de la queue des évènements de ce graphique.

Ne vous attendez donc pas à une mise à jour immédiate du graphique après un appel à des fonctions asynchrones. Utilisez la fonction ChartRedraw() pour forcer la mise à jour de l'apparence du graphique et de ses propriétés.

Fonction	Action
ChartApplyTemplate	Applique au graphique indiqué le modèle du fichier indiqué
ChartSaveTemplate	Enregistre les paramètres actuels du graphique dans un modèle avec le nom spécifié
ChartWindowFind	Retourne le numéro de sous- fenêtre, dans laquelle il y a un indicateur
ChartTimePriceToXY	Convertit les coordonnées du graphique de la représentation graphique du temps/prix pour les axes X et Y
ChartXYToTimePrice	Convertit les coordonnées X et Y du graphique en valeurs - le temps et le prix
ChartOpen	Ouvre un nouveau graphique avec le symbole indiqué et la période
ChartClose	Ferme le graphique indiqué
ChartFirst	Retourne l'identificateur du premier graphique du terminal de client
ChartNext	Retourne l'identificateur du graphique, suivant de l'indiqué
ChartSymbol	Retourne le nom du symbole du graphique indiqué
ChartPeriod	Retourne la valeur de la période du graphique indiqué
ChartRedraw	Appelle la copie forcée du graphique indiqué
ChartSetDouble	Spécifie la valeur du type double de la propriété correspondante du graphique indiqué
ChartSetInteger	Spécifie la valeur du type entier (datetime, int, color, bool ou char) de la propriété correspondante du graphique indiqué
ChartSetString	Spécifie la valeur du type string de la propriété correspondante du graphique indiqué
ChartGetDouble	Retourne la valeur de la propriété correspondante du graphique indiqué
ChartGetInteger	Retourne la valeur entière de la propriété correspondante du graphique indiqué

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
ChartGetString	Retourne la valeur de chaîne de la propriété correspondante du graphique indiqué
ChartNavigate	Réalise le décalage du graphique indiqué au nombre indiqué de barres par rapport à la position indiquée du graphique
ChartID	Retourne l'identificateur du graphique courant
ChartIndicatorAdd	Ajoute l'indicateur avec le handle indiqué sur la fenêtre indiquée du graphique
ChartIndicatorDelete	Supprime l'indicateur avec le nom spécifié de la fenêtre indiquée du graphique
ChartIndicatorGet	Retourne le handle de l'indicateur avec le nom court indiqué sur la fenêtre indiquée du graphique
ChartIndicatorName	Retourne le nom court de l'indicateur selon le numéro dans la liste des indicateurs sur la fenêtre indiquée du graphique
ChartIndicatorsTotal	Retourne le nombre de tous indicateurs qui sont attachés à la fenêtre indiquée du graphique
ChartWindowOnDropped	Retourne le numéro de sous-fenêtre du graphique, sur laquelle on jette l'expert donné, le script, l'objet ou l'indicateur par la souris
ChartPriceOnDropped	Retourne la coordonnée de prix correspondant au point, dans laquelle on jette par la souris l'expert donné ou le script
ChartTimeOnDropped	Retourne la coordonnée temporaire correspondant au point, dans laquelle on jette par la souris l'expert donné ou le script
ChartXOnDropped	Retourne la coordonnée selon l'axe X, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris
ChartYOnDropped	Retourne la coordonnée selon l'axe Y, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris
ChartSetSymbolPeriod	Change les valeurs du symbole et la période du graphique indiqué
ChartScreenShot	Fournit un screenshot du graphique de son état actuel dans un format de GIF, PNG ou BMP en fonction de l'extension spécifiée

© 2000-2024, MetaQuotes Ltd.

ChartApplyTemplate

Applique au graphique le modèle indiqué. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartApplyTemplate(
   long          chart_id,      // identificateur du graphique
   const string  filename       // nom du fichier avec le modèle
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

filename

[in] Le nom du fichier qui contient le modèle.

La valeur rendue

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Pour recevoir l'information sur l'erreur il faut appeler la fonction GetLastError().

Note

L'expert sera déchargé et ne pourra pas continuer le travail si un nouveau modèle sera chargé de l'expert sur le graphique à qui il est adjoint au moyen de cette fonction.

В целях безопасности право на торговлю при применении шаблона к графику может ограничиваться:

Права на торговлю не могут быть повышены при запуске советника путем применения шаблона с помощью функции ChartApplyTemplate().

Если у mql5-программы, которая вызывает функцию ChartApplyTemplate(), отсутствуют права на торговлю, то эксперт, загруженный при помощи шаблона, также не будет иметь прав на торговлю вне зависимости от настроек шаблона.

Если у mql5-программы, которая вызывает функцию ChartApplyTemplate(), есть права на торговлю, а в настройках шаблона права отсутствуют, то советник, загруженный при помощи шаблона, не будет иметь прав на торговлю.

Utilisation des modèles

Par les moyens du langage MQL5 on peut spécifier la multitude de propriétés du graphique, ainsi qu'établir les couleurs à l'aide de la fonction ChartSetInteger() :

- La couleur du fond du graphique;
- La couleur des axes, de l'échelle et de la ligne OHLC;
- La couleur de la grille;
- La couleur des volumes et des niveaux de l'ouverture des positions;
- La couleur de la barre en haut, de l'ombre et de l'encadrement du corps de la bougie d'haussier;

© 2000-2024, MetaQuotes Ltd.

- La couleur de la barre en bas, de l'ombre et de l'encadrement du corps de la bougie de baissier;
- La couleur de la ligne du graphique et des chandeliers japonais "Doji";
- La couleur du corps de la bougie d'haussier;
- La couleur du corps de la bougie de baissier;
- La couleur de la ligne du prix Bid;
- La couleur de la ligne du prix Ask;
- La couleur de la ligne du prix du dernier marché passé (Last);
- La couleur des niveaux des ordres Stop (Stop Loss et Take Profit).

En outre la multitudeed'objets graphiques et des indicateurspeut être sur le graphique. Il suffit de configurer une fois l'apparence du graphique avec tous les indicateurs nécessaires et le garder comme un modèle pour appliquer ce modèle à n'importe quel graphique plus tard.

La fonction ChartApplyTemplate () est destinée à être utilisé par le modèle précédemment gardé et peut être utilisée dans n'importe quel programme mql5. Le chemin vers le fichier qui contient le modèle est transmis comme le deuxième paramètre de la fonction ChartApplyTemplate (). est transmis le chemin vers le fichier qui contient le modèle. La recherche du fichier du modèle se réalise selon les règles suivantes:

- si au début du chemin il y a le séparateur - la barre oblique inversée "\" (est écrit comme "\\"), le modèle est recherché relativement le chemin - le répertoire_des données_du terminal\MQL5,
- s'il n'y a pas de la barre oblique inversée, alors le modèle est recherché par rapport le fichier EX5 exécutable où passe l'appel de la fonction (ChartApplyTemplate);
- si le modèle n'est pas trouvé dans les deux premiers variants, la recherche est effectuée dans un dossier le répertoire_du terminal\Profiles\Templates\.

Ici le répertoire_du terminal signifie le dossier à partir duquel a été lancé le terminal de client MetaTrader 5 et le répertoire_des données_du terminal signifie le dossier qui contient les fichiers modifiés et son emplacement peut dépendre du type du système d'exploitation et du nom d'utilisateur et des paramètres de sécurité de l'ordinateur. En général, ce sont les dossiers différents, bien que dans certains cas ils peuvent coïncider.

On peut savoir les emplacements des dossiers le répertoire_des données_du terminal et le répertoire_du terminal à l'aide de la fonction TerminalInfoString().

```
//--- le répertoire d'où est lancé le terminal
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
   Print("le répertoire du terminal:",terminal_path);
//--- le répertoire des données du terminal, dans lequel se trouve le dossier MQL5 ave
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   Print("Le répertoire des données du terminal:",terminal_data_path);
```

Les exemples de l'enregistrement:

```
//--- cherchons le modèle dans le dossier le répertoire_des données_du terminal\MQL5\
ChartApplyTemplate(0,"\\first_template.tpl"))
//--- cherchons le modèle dans le dossier le répertoire_du fichier_EX5_exécutable\, ap
ChartApplyTemplate(0,"second_template.tpl"))
//--- cherchons le modèle dans le dossier le répertoire_du fichier_EX5_exécutable\My_t
ChartApplyTemplate(0,"My_templates\\third_template.tpl"))
```

© 2000-2024, MetaQuotes Ltd.

Les modèles ne se rapportent pas aux ressources, on ne peut pas les insérer dans un fichier EX5 exécutable.

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- пример применения шаблона, расположенного в каталоге \MQL5\Files
   if(FileIsExist("my_template.tpl"))
     {
      Print("Шаблон my_template.tpl найден в каталоге \Files'");
      //--- применим найденный шаблон
      if(ChartApplyTemplate(0,"\\Files\\my_template.tpl"))
        {
         Print("Применили успешно шаблон 'my_template.tpl'");
         //--- принудительно перерисуем график для быстрого показа изменений
         ChartRedraw();
        }
      else
         Print("Не удалось применить шаблон 'my_template.tpl', ошибка ",GetLastError()
     }
   else
     {
      Print("Файл 'my_template.tpl' не найден в папке "
            +TerminalInfoString(TERMINAL_PATH)+"\\MQL5\\Files");
     }
  }
```

Voir aussi

Ressources

© 2000-2024, MetaQuotes Ltd.

ChartSaveTemplate

Garde les paramètres actuels de graphique dans le modèle avec le nom spécifié.

```
bool  ChartSaveTemplate(
   long          chart_id,      // l'identificateur du graphique
   const string  filename       // le nom du fichier pour la sauvegarde du modèle
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

filename

[in] Le nom du fichier pour la sauvegarde du modèle. L'extension ". tpl" sera ajouté au nom du fichier automatiquement, il ne faut pas le spécifier. Le modèle s'est gardé dans le dossier **le répertoire_du terminal**\Profiles\Templates\ et peut être utilisé pour l'application manuelle dans le terminal. Si le modèle avec ce nom existe déjà, son contenu sera enregistré de nouveau.

La valeur rendue

En cas de l'exécution successive la fonction rend true, autrement rend false. Pour recevoir l'information sur l'erreur il faut appeler la fonction GetLastError () .

Note

Le modèle vous permet de garder les paramètres du graphique avec tous ses indicateurs et les objets graphiques, pour l'appliquer sur un autre graphique.

Exemple:

```
//+------------------------------------------------------------------+
//|                                        Test_ChartSaveTemplate.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- input parameters
input string               symbol="GBPUSD";  // le symbole d'un nouveau graphique
input ENUM_TIMEFRAMES      period=PERIOD_H3; // le timeframe d'un nouveau graphique
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tout d'abord attachons les indicateurs au graphique
   int handle;
//---préparons l'indicateur à l'utilisation
```

```
   if(!PrepareZigzag(NULL,0,handle)) return; // la tentative a échoué, sortons
//--- attachons l'indicateur sur le graphique actuel, mais dans une fenêtre séparée
   if(!ChartIndicatorAdd(0,1,handle))
     {
      PrintFormat("On n'a pas réussi à ajouter %s/%s l'indicateur avec le handle =%d s
                  _Symbol,
                  EnumToString(_Period),
                  handle,
                  GetLastError());
      //--- cessons avant terme le travail du programme
      return;
     }
//--- ordonnons au graphique de mettre à jour pour voir tout de suite l'indicateur att
   ChartRedraw();
//--- trouvons deux dernières fractures du zigzag
   double two_values[];
   datetime two_times[];
   if(!GetLastTwoFractures(two_values,two_times,handle))
     {
      PrintFormat("On n'a pas réussi à trouver deux dernières fractures dans l'indicat
      //--- cessons avant terme le travail du programme
      return;
     }
//--- maintenant attachons le canal de la divergence standard
   string channel="StdDeviation Channel";
   if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
     {
      PrintFormat("On n'a pas réussi à créer l'objet %s. Code de l'erreur %d",
                  EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
      return;
     }
   else
     {
      //--- le canal est créé, définissons le deuxième point d'appui
      ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
      //--- spécifions le texte de l'infobulle au canal
      ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demo from MQL5 Help");
      //--- mettons à jour le graphique
      ChartRedraw();
     }
//--- sauvegardons le résultat dans un modèle
   ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- ouvrons un nouveau graphique et y attachons un modèle sauvegardé
   long new_chart=ChartOpen(symbol,period);
   //--- activons la démonstration des infobulles émergentes pour les objets graphique
   ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
   if(new_chart!=0)
     {
      //--- attachons un modèle sauvegardé sur un nouveau graphique
```

© 2000-2024, MetaQuotes Ltd.

```
      ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
     }
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Créé le handle du zigzag et assure la disponibilité de ses données |
//+------------------------------------------------------------------+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
  {
   ResetLastError();
//--- l'indicateur Zigzag doit être dans le dossier le répertoire_des données_du terml
   h=iCustom(sym,tf,"Examples\\Zigzag");
   if(h==INVALID_HANDLE)
     {
      PrintFormat("%s: On n'a pas réussi à créer le handle de l'indicateur Zigzag. Coc
                  __FUNCTION__,GetLastError());
      return false;
     }
//--- pendant la création du handle de l'indicateur dont il aura besoin de temps pour
   int k=0; // la quantité de tentatives pour attendre le calcul de l'indicateur
//--- attendons le calcul dans la boucle, faisons la pause de 50 millisecondes, si le
   while(BarsCalculated(h)<=0)
     {
      k++;
      //--- déduisons le nombre de tentatives
      PrintFormat("%s: k=%d",__FUNCTION__,k);
 //--- attendons 50 millisecondes, alors que l'indicateur sera calculé
      Sleep(50);
 //--- Si est fait plus de 100 tentatives, alors quelque chose ne va pas
      if(k>100)
        {
         //--- informons sur le problème
         PrintFormat("On n'a pas réussi à calculer l'indicateur pour%d de tentatives!"
         //--- cessons avant terme le travail du programme
         return false;
        }
     }
//--- tout est prêt, l'indicateur est créé et les valeurs sont calculées
   return true;
  }
//+------------------------------------------------------------------+
//| Cherche deux dernières fractures du zigzag et place aux tableaux |
//+------------------------------------------------------------------+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
  {
   double values[];         // le tableau pour la réception des valeurs du zigzag
   datetime times[];        // le tableau pour recevoir le temps
   int size=100;            // la taille des tableaux
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- copions les dernières 100 valeurs de l'indicateur
   int copied=CopyBuffer(handle,0,0,size,values);
//--- vérifions combien on a été copié
   if(copied<100)
     {
      PrintFormat("%s: On n'a pas réussi à copier %d valeurs de l'indicateur avec le h
                  __FUNCTION__,size,handle,GetLastError());
      return false;
     }
//--- définissons l'ordre de l'accès au tableau comme dans une série temporelle
   ArraySetAsSeries(values,true);
//--- inscrivons ici les numéros des barres, sur lesquelles sont trouvés les fractures
   int positions[];
//--- spécifions les tailles des tableaux
   ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//--- compteurs
   int i=0,k=0;
//--- commençons la recherche des fractures
   while(i<100)
     {
      double v=values[i];
      //--- les valeurs vides ne nous intéressent pas
      if(v!=0.0)
        {
         //--- rappelons le numéro de la barre
         positions[k]=i;
//--- rappelez la valeur du zigzag dans la fracture
         get_values[k]=values[i];
         PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
         //--- augmentons le compteur
         k++;
         //--- si on a trouvé deux fractures, nous cassons la boucle
         if(k>2) break;
        }
      i++;
     }
//--- définissons l'ordre de l'accès aux tableaux comme dans une série temporelle
   ArraySetAsSeries(times,true);   ArraySetAsSeries(get_times,true);
   if(CopyTime(_Symbol,_Period,0,size,times)<=0)
     {
      PrintFormat("%s: On n'a pas réussi à copier %d valeurs de CopyTime(). Code de l'
                  __FUNCTION__,size,GetLastError());
      return false;
     }
//--- trouvons le temps de l'ouverture des barres, sur lesquelles il y avait 2 dernièr
   get_times[0]=times[positions[1]];// la valeur avant-dernière sera inscrite par la p
   get_times[1]=times[positions[2]];// la troisième valeur de la fin sera la deuxième
   PrintFormat("%s: le premier=%s, le deuxième=%s",__FUNCTION__,TimeToString(get_times
//--- tout a réussi
```

© 2000-2024, MetaQuotes Ltd.

```
    return true;
  }
```

Voir aussi

ChartApplyTemplate(), Les ressources

© 2000-2024, MetaQuotes Ltd.

ChartWindowFind

Rend le numéro de sous- fenêtre, dans laquelle il y a un indicateur. Il y a 2 variantes de la fonction.

1. La fonction cherche sur le graphique indiqué la sous- fenêtre avec l'indicateur indiqué par "le nom court" (le nom court est affiché à gauche au dessus de la sous- fenêtre) et cas du succès rend le numéro de la sous- fenêtre.

```
int  ChartWindowFind(
   long     chart_id,                  // identificateur du graphique
   string   indicator_shortname        // nom court de l'indicateur, regarde INDICATOR
```

2. La fonction doit être appélée de l'indicateur d'utilisateur et récupère le numéro de la sous- fenêtre, où cet indicateur travaille.

```
int  ChartWindowFind();
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

indicator_shortname

[in] Le nom court de l'indicateur.

La valeur rendue

Le numéro du sous- fenêtre en cas du succès. Le zéro signifie une principale fenêtre du graphique. En cas de l'échec rend -1.

Note

Si la deuxième variante de la fonction est appelée (sans paramètres) du scipt ou l'expert, elle rend-1.

Il ne faut pas confondre un nom court et de l'indicateur et le nom du fichier qui est spécifié pendant la création de l'indicateur par les fonctionsiCustom() et IndicatorCreate(). Si le nom court de l'indicateur n'est pas spécifié explicitement, alors pendant la compilation le nom du fichier avec le code initial de l'indicateur y est spécifié.

Il est nécessaire de former correctement un nom court de l'indicateur, qui s'écrit dans la propriété INDICATOR_SHORTNAME à l'aide de la fonctionIndicatorSetString () . Nous recommandons que le nom court contienne les valeurs des paramètres d'entrée de l'indicateur, puisque l'identification de l'indicateur supprimé du graphique dans la fonctionChartIndicatorDelete()est faite selon le nom court.

Exemple:

```
#property script_show_inputs
//--- input parameters
input string   shortName="MACD(12,26,9)";
//+------------------------------------------------------------------+
//| Rend le numéro de la fenêtre du graphique avec l'indicateur indiqué |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
  {
   int window=-1;
//---
   if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
     {
      //--- fonction est appelée de l'indicateur, le nom n'est pas demandé
      window=ChartWindowFind();
     }
   else
     {
      //--- fonction est appelée de l'expert ou du script
      window=ChartWindowFind(0,short_name);
      if(window==-1) Print(__FUNCTION__+"(): Error = ",GetLastError());
     }
//---
   return(window);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int window=GetIndicatorSubWindowNumber(0,shortName);
   if(window!=-1)
      Print("Indicateur "+shortName+" se trouve dans la fenêtre #"+(string)window);
   else
      Print("Indicateur "+shortName+" n'a pas trouvé. window = "+(string)window);
  }
```

Voir aussi

ObjectCreate(), ObjectFind()

© 2000-2024, MetaQuotes Ltd.

ChartTimePriceToXY

Convertit les coordonnées du graphique de la représentation temps/prix pour les coordonnées selon axe X et Y.

```
bool  ChartTimePriceToXY(
   long           chart_id,      // l'identificateur du graphique
   int            sub_window,    //le numéro de la sous-fenêtre
   datetime       time,          // le temps sur le graphique
   double         price,         // le temps sur le graphique
   int&           x,             // la coordonnée X pour le temps sur le graphique
   int&           y              // la coordonnée Y pour le prix sur le graphique
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

time

[in] La signification du temps sur le graphique, pour laquelle sera reçue la valeur en pixels sur l'axe X. Le début des coordonnées se trouve dans un angle gauche supérieur de la fenêtre principale du graphique.

price

[in] La signification du prix sur le graphique, pour laquelle sera reçue la valeur en pixels sur l'axe Y. Le début des coordonnées se trouve dans un angle gauche supérieur de la fenêtre principale du graphique.

x

[out] La variable, où sera reçue la transformation du temps en coordonnée X.

y

[out] La variable, où sera reçue la transformation du prix en coordonnée Y.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement - false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError () .

Voir aussi

ChartXYToTimePrice()

© 2000-2024, MetaQuotes Ltd.

ChartXYToTimePrice

Convertit les coordonnées X et Y du graphique en valeurs le temps et le prix.

```
bool  ChartXYToTimePrice(
   long           chart_id,      // l'identificateur du graphique
   int            x,             // la coordonnée X sur le graphique
   int            y,             // la coordonnée Y sur le graphique
   int&           sub_window,    // le numéro de la sous-fenêtre
   datetime&      time,          // le temps sur le graphique
   double&        price          // le prix sur le graphique
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

x

[in] La coordonnée X.

y

[in] La coordonnée Y.

sub_window

[out] La variable, où sera inscrit le numéro de la sous- fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

time

[out] La signification du temps sur le graphique, pour laquelle sera reçue la valeur en pixels sur l'axe X. Le début des coordonnées se trouve dans un angle gauche supérieur de la fenêtre principale du graphique.

price

[out] La signification du prix sur le graphique, pour laquelle sera reçue la valeur en pixels sur l'axe Y. Le début des coordonnées se trouve dans un angle gauche supérieur de la fenêtre principale du graphique.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement - false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError () .

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- déduisons les paramètres de l'événement sur le graphique
   Comment(__FUNCTION__,": id=",id," lparam=",lparam," dparam=",dparam," sparam=",spa
//--- Si ce sont les événements du clic de la souris sur le graphique
   if(id==CHARTEVENT_CLICK)
     {
      //--- préparons les variables
      int      x     =(int)lparam;
      int      y     =(int)dparam;
      datetime dt    =0;
      double   price =0;
      int      window=0;
      //--- transformons les coordonnées X et Y dans les termes la date/temps
      if(ChartXYToTimePrice(0,x,y,window,dt,price))
        {
         PrintFormat("Window=%d X=%d  Y=%d  =>  Time=%s  Price=%G",window,x,y,TimeToSt
         //--- faisons la transformation inverse: (X,Y) => (Time,Price)
         if(ChartTimePriceToXY(0,window,dt,price,x,y))
            PrintFormat("Time=%s  Price=%G  =>  X=%d  Y=%d",TimeToString(dt),price,x,y
         else
            Print("ChartTimePriceToXY return error code: ",GetLastError());
         //--- delete lines
         ObjectDelete(0,"V Line");
         ObjectDelete(0,"H Line");
         //--- create horizontal and vertical lines of the crosshair
         ObjectCreate(0,"H Line",OBJ_HLINE,window,dt,price);
         ObjectCreate(0,"V Line",OBJ_VLINE,window,dt,price);
         ChartRedraw(0);
        }
      else
         Print("ChartXYToTimePrice return error code: ",GetLastError());
      Print("+------------------------------------------------------------------+");
     }
  }
```

Voir aussi

ChartTimePriceToXY()

© 2000-2024, MetaQuotes Ltd.

ChartOpen

Ouvre un nouveau graphique avec le symbole indiqué et la période.

```
long  ChartOpen(
   string           symbol,      // nom du symbole
   ENUM_TIMEFRAMES  period       // période
   );
```

Paramètres

symbol

[in] Le symbole du graphique. NULL signifit le symbole du graphique courant (auquel l'expert est attaché).

period

[in] La période du graphique (le temps trame). Peut prendre une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 est la période du graphique courant.

La valeur rendue

A l'ouverture réussie du graphique la fonction rend l'identificateur du graphique. Autrement rend 0.

Note

Le nombre possible maximum de graphiques simultanément ouverts dans le terminal ne peut pas ne peut pas excéder la valeur CHARTS_MAX.

© 2000-2024, MetaQuotes Ltd.

ChartFirst

Rend l'identificateur du premier graphique du terminal de client.

```
long  ChartFirst();
```

La valeur rendue

L'identificateur du graphique.

© 2000-2024, MetaQuotes Ltd.

ChartNext

La fonction rend l'identificateur du graphique, suivant de l'indiqué.

```
long  ChartNext(
   long  chart_id      // identificateur du graphique
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 ne signifie pas le graphique courant. 0 signifie "rendre l'identificateur du premier graphique".

La valeur rendue

L'identificateur du graphique. Si la liste des graphiques s'est achevé, la fonction rend -1.

Exemple:

```
//--- variables pour les identificateurs des graphiques
   long currChart,prevChart=ChartFirst();
   int i=0,limit=100;
   Print("ChartFirst = ",ChartSymbol(prevChart)," ID = ",prevChart);
   while(i<limit)// nous avons certainement pas plus de 100 graphiques ouverts
     {
      currChart=ChartNext(prevChart); // nous recevons un nouveau graphique à la base
      if(currChart<0) break;          // ont atteint la fin de la liste des graphiques
      Print(i,ChartSymbol(currChart)," ID = ",currChart);
      prevChart=currChart;// sauvegardons  l'identificateur du graphique courant pour
      i++;// n'oublions pas d'augmenter le compteur
     }
```

© 2000-2024, MetaQuotes Ltd.

ChartClose

Ferme le graphique indiqué.

```
bool  ChartClose(
   long  chart_id=0      // identificateur du graphique
   );
```

Paramètres

chart_id=0

[in] L'identificateur du graphique. 0 signifit le graphique courant.

La valeur rendue

Rend true en cas du succès, autrement rend false.

© 2000-2024, MetaQuotes Ltd.

ChartSymbol

Rend le nom du symbole du graphique indiqué.

```
string  ChartSymbol(
   long  chart_id=0      // identificateur du graphique
   );
```

Paramètres

chart_id=0

[in] L'identificateur du graphique. 0 signifit le graphique courant.

La valeur rendue

Si le graphique n'existe pas, la chaîne vide se rend.

Voir aussi

ChartSetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

ChartPeriod

Rend la valeur de la période di graphique indiqué.

```
ENUM_TIMEFRAMES  ChartPeriod(
   long  chart_id=0      // identificateur du graphique
   );
```

Paramètres

chart_id=0

[in] L'identificateur du graphique. 0 signifit le graphique courant.

La valeur rendue

La valeur du type ENUM_TIMEFRAMES. Si le graphique n'existe pas, revient 0.

© 2000-2024, MetaQuotes Ltd.

ChartRedraw

Appelle la copie forcée du graphique indiqué.

```
void  ChartRedraw(
   long  chart_id=0      // identificateur du graphique
   );
```

Paramètres

chart_id=0

[in] L'identificateur du graphique. 0 signifit le graphique courant.

Note

Est appliqué d'habitude après le changement des propriétés des objets.

Voir aussi

Les objets graphiques

© 2000-2024, MetaQuotes Ltd.

ChartSetDouble

Spécifie la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être du type double. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartSetDouble(
   long                          chart_id,      // identificateur du graphique
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,       // identificateur de la propriété
   double                        value          // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_DOUBLE (sauf les propriétés read-only).

value

[in] La valeur de la propriété.

La valeur rendue

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError().

Note

La fonction est asynchrone, ce qui signifie que la fonction n'attends pas l'exécution de la commande qui a été ajoutée avec succès dans la queue d'évènements du graphique spécifié. Au lieu de cela, elle retourne le contrôle immédiatement. La propriété ne sera changée qu'après la gestion de la commande correspondante de la queue du graphique. Pour exécuter immédiatement les commandes de la file d'attente du graphique, appelez la fonction ChartRedraw.

Si vous souhaitez changer immédiatement plusieurs propriétés d'un graphique en une seule fois, les fonctions correspondantes (ChartSetString, ChartSetDouble, ChartSetString) doivent être exécutées en un seul bloc de code, après lequel vous devrez appeler ChartRedraw une seule fois.

Pour vérifier le résultat de l'exécution de la commande, vous pouvez utiliser une fonction qui retourne la propriété correspondante du graphique (ChartGetInteger, ChartGetDouble, ChartSetString). Notez cependant que ces fonctions sont synchrone et attendent le résultat de l'exécution.

© 2000-2024, MetaQuotes Ltd.

ChartSetInteger

Spécifie la valeur de la propriété correspondante du graphique indiqué. Lapropriété du graphique doit être du type datetime, int, color, bool ou char. La commande donnée entre dans la file d'attente des messages du graphique et n'est exécutée qu'après le traitement de toutes les commandes précédentes

```
bool  ChartSetInteger(
   long                          chart_id,      // identificateur du graphique
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // identificateur de la propriété
   long                          value          // valeur
   );
```

```
bool  ChartSetInteger(
   long                          chart_id,      // identificateur du graphique
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // identificateur de la propriété
   int                           sub_window,    // numéro de la sous-fenêtre
   long                          value          // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_INTEGER (sauf les propriétés read-only).

sub_window

[in] Le numéro de la sous-fenêtre du graphique. Pour la première variante la valeur est égale à 0 par défaut (la fenêtre principale du graphique). La plupart des propriétés ne demandent pas l'indication du numéro de la sous-fenêtre.

value

[in] La valeur de la propriété.

Valeur de Retour

Renvoie true si la commande est correctement placée dans la file d'attente graphique, sinon false. Pour obtenir plus d'informations sur l'erreur, il faut appeler la fonction GetLastError().

Note

La fonction est asynchrone, ce qui signifie que la fonction n'attends pas l'exécution de la commande qui a été ajoutée avec succès dans la queue d'évènements du graphique spécifié. Au lieu de cela, elle retourne le contrôle immédiatement. La propriété ne sera changée qu'après la gestion de la commande correspondante de la queue du graphique. Pour exécuter immédiatement les commandes de la file d'attente du graphique, appelez la fonction ChartRedraw.

© 2000-2024, MetaQuotes Ltd.

Si vous souhaitez changer immédiatement plusieurs propriétés d'un graphique en une seule fois, les fonctions correspondantes (ChartSetString, ChartSetDouble, ChartSetString) doivent être exécutées en un seul bloc de code, après lequel vous devrez appeler ChartRedraw une seule fois.

Pour vérifier le résultat de l'exécution de la commande, vous pouvez utiliser une fonction qui retourne la propriété correspondante du graphique (ChartGetInteger, ChartGetDouble, ChartSetString). Notez cependant que ces fonctions sont synchrone et attendent le résultat de l'exécution.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Activation des évènements des mouvements de la souris sur la fenêtre du graphiqu
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- La mise à jour forcée des propriétés du graphique assure qu'elles seront prises
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // Bouton gauche de la souris
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // Bouton droit de la souris
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // Bouton du milieu de la souris
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // Coordonnée X de la souris
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // Coordonnée Y de la souris
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // Touche shift
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // Touche controle
   return(res);
  }
//+------------------------------------------------------------------+
//| Fonction ChartEvent                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

© 2000-2024, MetaQuotes Ltd.

ChartSetString

Spécifie la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être du type string. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartSetString(
   long                          chart_id,      // identificateur du graphique
   ENUM_CHART_PROPERTY_STRING    prop_id,       // identificateur de la propriété
   string                        str_value      // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_STRING (sauf les propriétés read-only).

str_value

[in] La chaîne pour l'installation de la propriété. La longueur de la chaîne ne peut pas excéder 2045 caractères (les caractères superflus seront coupés).

La valeur rendue

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError().

Note

La fonction ChartSetString peut être utilisée pour la sortie des commentaires sur le graphique au lieu de la fonction Comment.

La fonction est asynchrone, ce qui signifie que la fonction n'attends pas l'exécution de la commande qui a été ajoutée avec succès dans la queue d'évènements du graphique spécifié. Au lieu de cela, elle retourne le contrôle immédiatement. La propriété ne sera changée qu'après la gestion de la commande correspondante de la queue du graphique. Pour exécuter immédiatement les commandes de la file d'attente du graphique, appelez la fonction ChartRedraw.

Si vous souhaitez changer immédiatement plusieurs propriétés d'un graphique en une seule fois, les fonctions correspondantes (ChartSetString, ChartSetDouble, ChartSetString) doivent être exécutées en un seul bloc de code, après lequel vous devrez appeler ChartRedraw une seule fois.

Pour vérifier le résultat de l'exécution de la commande, vous pouvez utiliser une fonction qui retourne la propriété correspondante du graphique (ChartGetInteger, ChartGetDouble, ChartSetString). Notez cependant que ces fonctions sont synchrone et attendent le résultat de l'exécution

Exemple:

```
void OnTick()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
   string comment=StringFormat("Выводим цены:\nAsk = %G\nBid = %G\nSpread = %d",
                               Ask,Bid,Spread);
   ChartSetString(0,CHART_COMMENT,comment);
  }
```

Voir aussi

Comment, ChartGetString

© 2000-2024, MetaQuotes Ltd.

ChartGetDouble

Retourne la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être du type double. Il y a 2 variantes de la fonction.

1. Retourne directement la valeur de la propriété.

```
double  ChartGetDouble(
   long                          chart_id,         // identificateur du graphique
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,          // identificateur de la propriété
   int                           sub_window=0      // numéro de la sous-fenêtre, si c'es
   );
```

2. Retourne true ou false en fonction du succès de l'exécution de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable transmise par référence comme dernier paramètre.

```
bool  ChartGetDouble(
   long                          chart_id,         // identificateur du graphique
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,          // identificateur de la propriété
   int                           sub_window,       // numéro de la sous-fenêtre
   double&                       double_var        // valeur de la propriété
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être l'une des valeurs de l'énumération ENUM_CHART_PROPERTY_DOUBLE.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. Pour la première variante la valeur est égale à 0 par défaut (la fenêtre principale du graphique). La plupart des propriétés ne demandent pas l'indication du numéro de la sous-fenêtre.

double_var

[out] La variable du type double, acceptant la valeur de la propriété demandée.

Valeur de Retour

Une valeur de type double.

Pour la deuxième variante de l'appel, retourne true si la propriété donnée est supportée et que la valeur a été placée dans la variable double_var, sinon retourne false. Pour obtenir plus d'informations sur l'erreur, il est nécessaire d' appeler la fonction GetLastError().

Note

La fonction est synchrone, ce qui signifie qu'elle attend l'exécution de toutes les commandes qui ont été ajoutées dans la queue des évènements du graphique avant son appel.

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
   double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,0);
   double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,0);
   Print("CHART_PRICE_MIN = ",priceMin);
   Print("CHART_PRICE_MAX = ",priceMax);
  }
```

© 2000-2024, MetaQuotes Ltd.

ChartGetInteger

Rend la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être des types datetime, int ou bool. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
long  ChartGetInteger(
   long                          chart_id,         // identificateur du graphique
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // identificateur de la propriété
   int                           sub_window=0      // numéro de la sous-fenêtre, si c'e
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  ChartGetInteger(
   long                          chart_id,         // identificateur du graphique
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // identificateur de la propriété
   int                           sub_window,       // numéro de la sous-fenêtre
   long&                         long_var          // acceptons ici la valeur de la pro
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_INTEGER.

sub_window

[in] Pour la première variante la valeur est égale à 0 par défaut (une fenêtre principale du graphique). La plupart des propriétés ne demandent pas l'indication du numéro de la sous-fenêtre.

long_var

[out] La valeur du type long, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type long.

Pour la deuxième variante de l'appel rend true, si la propriété donnée est supportée et la valeur était placé à la variable long_var, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d' appeler la fonction GetLastError().

Note

La fonction est synchrone, ce qui signifie qu'elle attend l'exécution de toutes les commandes qui ont été ajoutées dans la queue des évènements du graphique avant son appel.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
   int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   Print("CHART_HEIGHT_IN_PIXELS = ",height," pixels");
   Print("CHART_WIDTH_IN_PIXELS = ",width," pixels");
  }
```

© 2000-2024, MetaQuotes Ltd.

ChartGetString

Rend la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être du type string. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
string  ChartGetString(
   long                          chart_id,         // identificateur du graphique
   ENUM_CHART_PROPERTY_STRING    prop_id           // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  ChartGetString(
   long                          chart_id,         // identificateur du graphique
   ENUM_CHART_PROPERTY_STRING    prop_id,          // identificateur de la propriété
   string&                       string_var        // acceptons ici la valeur de la prop
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_STRING.

string_var

[out] La variable du type string, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type string.

Pour la deuxième variante de l'appel rend true, si la propriété donnée est supportée et la valeur était placé à la variable string_var, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d' appeler la fonction GetLastError().

Note

La fonction ChartGetString peut être utilisée pour la lecture des commentaires affichés sur le graphique par les fonctions Comment ou ChartSetString.

La fonction est synchrone, ce qui signifie qu'elle attend l'exécution de toutes les commandes qui ont été ajoutées dans la queue des évènements du graphique avant son appel.

Exemple:

```
void OnStart()
  {
   ChartSetString(0,CHART_COMMENT,"Test comment.\nSecond line.\nThird!");
```

© 2000-2024, MetaQuotes Ltd.

```
  ChartRedraw();
  Sleep(1000);
  string comm=ChartGetString(0,CHART_COMMENT);
  Print(comm);
 }
```

Voir aussi

Comment, ChartSetString

© 2000-2024, MetaQuotes Ltd.

ChartNavigate

Réalise le décalage du graphique indiqué au nombre indiqué de barres par rapport à la position indiquée du graphique.

```
bool  ChartNavigate(
   long                  chart_id,      // identificateur du graphique
   ENUM_CHART_POSITION   position,      // position
   int                   shift=0        // valeur du décalage
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

position

[in] La position de graphique pour exécuter le décalage. La valeur peut être une des valeurs de l'énumération ENUM_CHART_POSITION.

shift=0

[in] Le nombre de barres, auxquels il faut déplacer le graphique. La valeur positive signifie le décalage à droite (à la fin du graphique), la valeur négative signifie le décalage à gauche (vers le début du graphique). Le décalage nul est justifié, quand on produit la navigation vers le début ou la fin du graphique.

La valeur rendue

Rend true en cas du succès, autrement rend false.

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- recevons le handle de l'indicateur actuel
   long handle=ChartID();
   string comm="";
   if(handle>0) // en cas de succès, configurons supplémentairement
     {
      //--- désactivons l'autodéfilement
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- établissons l'alinéa du bord droit du graphique
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- affichons en forme des bougies
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- établir le mode de l'affichage des volumes de tick
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- préparons le texte pour la sortie dans Comment()
comm="Faisons défiler 10 barres à droite du début de l'histore";
//--- déduisons le commentaire
Comment(comm);
//--- faisons défiler 10 barres à droite du début de l'histore
ChartNavigate(handle,CHART_BEGIN,10);
//--- recevons le numéro de la première barre affichée sur le graphique (le nume
long first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
//--- ajoutons le symbole du transfert de la ligne
comm=comm+"\r\n";
//--- completons le commentaire
comm=comm+"La première barre sur le graphique a le numéro"+IntegerToString(first
//--- déduisons le commentaire
Comment(comm);
//--- attendons 5 secondes pour avoir le temps de voir comment avance le graphic
Sleep(5000);

//--- écrivons le texte du commentaire
comm=comm+"\r\n"+"Faisons défiler 10 barres à gauche du bord droit du graphique"
Comment(comm);
//--- faisons défiler 10 barres à gauche du bord droit du graphique
ChartNavigate(handle,CHART_END,-10);
//--- recevons le numéro de la première barre affichée sur le graphique (le nume
first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
comm=comm+"\r\n";
comm=comm+"La première barre sur le graphique a le numéro"+IntegerToString(first
Comment(comm);
//--- attendons 5 secondes pour avoir le temps de voir comment avance le graphic
Sleep(5000);

//--- un nouveau bloc du défilement du graphique
comm=comm+"\r\n"+"Faisons défiler 300 barres à droite du début de l'histore";
Comment(comm);
//--- faisons défiler 300 barres à droite du début de l'histore
ChartNavigate(handle,CHART_BEGIN,300);
first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
comm=comm+"\r\n";
comm=comm+"La première barre sur le graphique a le numéro"+IntegerToString(first
Comment(comm);
//--- attendons 5 secondes pour avoir le temps de voir comment avance le graphic
Sleep(5000);

//--- un nouveau bloc du défilement du graphique
comm=comm+"\r\n"+"Faisons défiler 300 barres à gauche du bord droit du graphique
Comment(comm);
//--- faisons défiler 300 barres à gauche du bord droit du graphique
ChartNavigate(handle,CHART_END,-300);
first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
comm=comm+"\r\n";
```

© 2000-2024, MetaQuotes Ltd.

```
      comm=comm+"La première barre sur le graphique a le numéro"+IntegerToString(first
      Comment(comm);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

ChartID

Rend l'identificateur du graphique courant.

```
long  ChartID();
```

La valeur rendue

La valeur du type long.

© 2000-2024, MetaQuotes Ltd.

ChartIndicatorAdd

Ajoute l'indicateur avec le handle indiqué sur la fenêtre indiquée du graphique. L'indicateur et le graphique doivent être construits sur le même symbole et le temps trame.

```
bool  ChartIndicatorAdd(
   long  chart_id,                 // l'identificateur du graphique
   int   sub_window                // le numéro de la sous-fenêtre
   int   indicator_handle          //le handle indiqué
   );
```

Les paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie une fenêtre principale du graphique. Pour ajouter un indicateur dans une nouvelle fenêtre, le paramètre doit être plus grand que l'index de la dernière fenêtre existante, alors il doit être égal à CHART_WINDOWS_TOTAL. Si la valeur du paramètre dépasse la valeur CHART_WINDOWS_TOTAL, alors la nouvelle fenêtre ne sera pas créée et l'indicateur ne sera pas ajouté.

indicator_handle

[in] Le handle de l'indicateur.

La valeur rendue

Rend true en cas du succès, autrement rend false. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError(). L'erreur 4114 signifie que le graphique et l'indicateur ajouté se distinguent selon le symbole ou le temps trame.

Note

Si sur une principale fenêtre du graphique on ajoute l'indicateur, qui doit être dessiné dans le sous-fenêtre séparé (par exemple, iMACD inséré ou l'indicateur d'utilisateur avec la propriété indiquée #property indicator_separate_window), un tel indicateur peut être invisible, bien qu'il sera dans la liste des indicateurs. Cela signifie que l'échelle de l'indicateur donné se distingue de l'échelle du graphique des prix et les valeurs de l'indicateur ajouté ne se sont pas introduits dans la gamme affichée du graphique de prix. Dans ce cas GetLastError() rendra le code nul signifiant l'absence de l'erreur. On pourra observer les valeurs d'un tel indicateur "invisible" dans "Fenêtre des données" et recevoir des autres programmes MQL5.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
#property description "L'expert pour la démonstration du travail avec la fonction Char
#property description "Après le lancement sur le graphique (et la réception de l'erreu
#property description "les propriétés du conseiller et spécifiez les paramètres justes
#property description "L'indicateur MACD sera ajouté sur le graphique."

//--- input parameters
input string          symbol="AUDUSD";     // le nom du symbole
input ENUM_TIMEFRAMES period=PERIOD_M12;  // le temps trame
input int    fast_ema_period=12;          // la période de la moyenne rapide MACD
input int    slow_ema_period=26;          // la période de la moyenne lente MACD
input int      signal_period=9;           // la période de la prise de moyenne de la c
input ENUM_APPLIED_PRICE apr=PRICE_CLOSE; // le type du prix pour le calcul MACD

int indicator_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_period,
//--- essayons ajouter l'indicateur sur le graphique
   if(!AddIndicator())
     {
      //--- la fonction AddIndicator () a refusé d'ajouter l'indicateur sur le graphic
      int answer=MessageBox("Quand même, essayer d'ajouter MACD sur le graphique?",
                            "Le symbole et/ou le temps trame sont spécifiés incorrecte
                            MB_YESNO // les boutons du choix "Yes" et "No" seront mont
                            );
      //--- si l'utilisateur insiste en tout cas sur l'utilisation incorrecte de Chart
      if(answer==IDYES)
        {
         //--- d'abord communiquons sur cela au journal
         PrintFormat("L'attention! %s: Essayons ajouter l'indicateur MACD(%s/%s) sur l
                     __FUNCTION__,symbol,EnumToString(period),_Symbol,EnumToString(_Pe
         //--- recevons le numéro du nouveau sous-fenêtre, où tentons d'ajouter l'indi
         int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
         //--- maintenant faisons la tentative condamnée à l'erreur
         if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
            PrintFormat("On n'a pas réussi à ajouter l'indicateur MACD sur la fenêtre
                        subwindow,GetLastError());
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
// l'expert ne fait rien
  }
//+------------------------------------------------------------------+
//| La fonction de la vérification et le supplément de l'indicateur sur le graphique
//+------------------------------------------------------------------+
bool AddIndicator()
  {
//--- le message déduit
   string message;
//--- vérifions sur la coïncidence le symbole de l'indicateur et le symbole du graphic
```

© 2000-2024, MetaQuotes Ltd.

```
   if(symbol!=_Symbol)
     {
      message="La démonstration de l'utilisation de la fonction Demo_ChartIndicatorAdd
      message=message+"\r\n";
      message=message+"On ne peut pas ajouter sur le graphique l'indicateur calculé su
      message=message+"\r\n";
      message=message+"Indiquez le symbole du graphique dans les propriétés de l'exper
      Alert(message);
      //--- la sortie anticipé, n'ajoutons pas l'indicateur sur le graphique
      return false;
     }
//--- vérifions sur la coïncidence le temps trame de l'indicateur et le temps trame du
   if(period!=_Period)
     {
      message="On ne peut pas ajouter sur le graphique l'indicateur, calculé sur un au
      message=message+"\r\n";
      message=message+"Indiquez dans les propriétés de l'expert le temps trame, qui es
      Alert(message);
      //--- la sortie anticipé, n'ajoutons pas l'indicateur sur le graphique
      return false;
     }
//--- toutes les vérificatios ont passé, le symbole et la période de l'indicateur cor
   if(indicator_handle==INVALID_HANDLE)
     {
      Print(__FUNCTION__," Créons l'indicateur MACD");
      indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_peri
      if(indicator_handle==INVALID_HANDLE)
        {
         Print("On n'a pas réussi à créer l'indicateur MACD. Le code de l'erreur ",Get
        }
     }
//--- obliterons le code de l'erreur
   ResetLastError();
//---appliquons l'indicateur au graphique
   Print(__FUNCTION__," Ajoutons l'indicateur MACD sur le graphique");
   Print("MACD est construit sur ",symbol,"/",EnumToString(period));
//--- recevons le numéro du nouveau sous-fenêtre, où ajoutons l'indicateur MACD
   int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   PrintFormat("Ajoutons l'indicateur MACD ) à la fenêtre du graphique %d ",subwindow)
   if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
     {
      PrintFormat("On n'a pas réussi à ajouter l'indicateur MACD sur la fenêtre du gra
                  subwindow,GetLastError());
     }
//--- le supplément de l'indicateur sur le graphique a passé avec succès
   return(true);
  }
```

Voir aussi

ChartIndicatorDelete(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate()

© 2000-2024, MetaQuotes Ltd.

ChartIndicatorDelete

Supprime l'indicateur avec le nom spécifié de la fenêtre indiquée du graphique.

```
bool  ChartIndicatorDelete(
   long           chart_id,              // l'identificateur du graphique
   int            sub_window             // le numéro de la sous-fenêtre
   const string   indicator_shortname    // le nom court de l'indicateur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

const indicator_shortname

[in] Le nom court de l'indicateur, qui est spécifié dans la propriété INDICATOR_SHORTNAME par la fonction IndicatorSetString(). On peut recevoir le nom court de l'indicateur par la fonction ChartIndicatorName().

La valeur rendue

Rend true si l'indicateur a été supprimé avec succès, autrement rend false. Pour recevoir l'information sur l'erreur il faut appeler la fonction GetLastError () .

Note

S'il y a plusieurs indicateurs avec le même nom court dans la sous-fenêtre spécifiée du graphique, sera supprimé le premier indicateur.

Si les autres indicateurs sont construits sur les valeurs de l'indicateur supprimé sur le même graphique, ils seront également supprimés.

Il ne faut pas confondre un nom court et de l'indicateur et le nom du fichier qui est spécifié pendant la création de l'indicateur par les fonctions iCustom() et IndicatorCreate(). Si le nom court de l'indicateur n'est pas spécifié explicitement, alors pendant la compilation le nom du fichier avec le code initial de l'indicateur y est spécifié.

La suppression de l'indicateur du graphique ne signifie pas que la partie calculé de l'indicateur sera aussi supprimée de la mémoire du terminal. Pour vider le handle de l'indicateur utiliser la fonction IndicatorRelease().

Il est nécessaire de former correctement un nom court de l'indicateur, qui s'écrit dans la propriété INDICATOR_SHORTNAME à l'aide de la fonction IndicatorSetString () . Nous recommandons que le nom court contienne les valeurs des paramètres d'entrée de l'indicateur, puisque l'identification de l'indicateur supprimé du graphique dans la fonction ChartIndicatorDelete() est faite selon le nom court.

L'exemple de la suppression de l'indicateur pendant la mauvaise initialisation:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                   Demo_ChartIndicatorDelete.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      first_param=1;
input int      second_param=2;
input int      third_param=3;
input bool     wrong_init=true;
//--- indicator buffers
double         HistogramBuffer[];
string         shortname;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int res=INIT_SUCCEEDED;
//--- attachons le tableau HistogramBuffer au tampon d'indicateur
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- construisons le nom court de l'indicateur à la base des paramètres d'entrée
   shortname=StringFormat("Demo_ChartIndicatorDelete(%d,%d,%d)",
                          first_param,second_param,third_param);
   IndicatorSetString(INDICATOR_SHORTNAME,shortname);
//--- si on a spécifié l'achèvement forcé de l'indicateur, rendons la valeur non nulle
   if(wrong_init) res=INIT_FAILED;
   return(res);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- la position initiale pour le travail dans la boucle
   int start=prev_calculated-1;
   if(start<0) start=0;
//--- remplissons le tampon d'indicateur par les valeurs
   for(int i=start;i<rates_total;i++)
     {
```

```
      HistogramBuffer[i]=close[i];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| gestionnaire d'événements Deinit                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   PrintFormat("%s: le code de la raison de la désinitialisation =%d",__FUNCTION__,rea
   if(reason==REASON_INITFAILED)
     {
      PrintFormat("L'indicateur avec le nom court %s (le fichier %s) se supprime du gr
      int window=ChartWindowFind();
      bool res=ChartIndicatorDelete(0,window,shortname);
      //--- analysons le résultat de l'appel ChartIndicatorDelete()
      if(!res)
        {
         PrintFormat("On n'a pas réussi à supprimer l'indicateur %s de la fenêtre #%d.
                     shortname,window,GetLastError());
        }
     }
  }
```

Voir aussi

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2024, MetaQuotes Ltd.

ChartIndicatorGet

Rend le handle de l'indicateur avec le nom court indiqué sur la fenêtre indiquée du graphique.

```
int  ChartIndicatorGet(
   long           chart_id,              // l'identificateur du graphique
   int            sub_window             // le numéro de la sous-fenêtre
   const string   indicator_shortname    // le nom court de l'indicateur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

const indicator_shortname

[in] Le nom court de l'indicateur, qui est spécifié dans la propriété INDICATOR_SHORTNAME par la fonction IndicatorSetString(). On peut recevoir le nom court de l'indicateur par la fonction ChartIndicatorName().

La valeur rendue

Rend le handle de l'indicateur en cas de l'exécution réussie, autrement INVALID_HANDLE. Pour recevoir l'informationl'erreur, il faut appeler la fonctionGetLastError().

Note

Le descripteur de l'indicateur obtenu à l'aide de la fonction ChartIndicatorGet() augmente le compteur interne d'utilisation de l'indicateur. Le système d'exécution du terminal conserve en mémoire tous les indicateurs dont le compteur est supérieur à zéro. Par conséquent, le descripteur d'indicateur qui n'est plus nécessaire doit être libéré immédiatement et explicitement en utilisant IndicatorRelease() dans le même programme, comme illustré dans l'exemple ci-dessous. Sinon, il sera impossible de trouver le descripteur "abandonné" et de le libérer correctement depuis un autre programme.

Il est nécessaire de former correctement un nom court de l'indicateur à sa création, qui à l'aide de la fonction IndicatorSetString() est enregistré à la propriétéINDICATOR_SHORTNAME. Nous recommandons que le nom court contienne les valeurs des paramètres d'entrée de l'indicateur, puisque l'identification de l'indicateur dans la fonction ChartIndicatorGet() est faite selon le nom court.

Un autre moyen de l'identification de l'indicateur - recevoir la liste de ses paramètres selon le handle donné à l'aide de la fonction IndicatorParameters() et puis analyser les valeurs reçues.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- le nombre de fenêtres sur le graphique (il y a toujours au moins une fenêtre
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   //--- passons par les fenêtres
   for(int w=0;w<windows;w++)
     {
      //--- combien d'indicateurs dans cette fenêtre/sous-fenêtre
      int total=ChartIndicatorsTotal(0,w);
      //---trions tous les indicateurs dans la fenêtre
      for(int i=0;i<total;i++)
        {
         //--- recevons le nom court de l'indicateur
         string name=ChartIndicatorName(0,w,i);
         //--- recevons le handle de l'indicateur
         int handle=ChartIndicatorGet(0,w,name);
         //--- déduisons dans un journal
         PrintFormat("Window=%d,  index=%d,  name=%s,  handle=%d",w,i,name,handle);
         //--- обязательно освобождаем хендл индикатора, как только он становится не н
         IndicatorRelease(handle);
        }
     }
  }
```

Voir aussi

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), IndicatorParameters()

© 2000-2024, MetaQuotes Ltd.

ChartIndicatorName

Rend le nom court de l'indicateur selon le numéro dans la liste des indicateurs sur la fenêtre indiquée du graphique.

```
string  ChartIndicatorName(
   long  chart_id,       // l'identificateur du graphique
   int   sub_window      // le numéro de la sous-fenêtre
   int   index           // L'index de l'indicateur dans la liste des indicateurs ajout
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

index

[in] L'index d'un indicateur dans la liste des indicateurs. Le numérotage des indicateurs commence par le zéro, c'est-à-dire le premier indicateur dans la liste a l'index nul. On peut recevoir le nombre d'indicateurs dans la liste par la fonctionChartIndicatorsTotal().

La valeur rendue

Le nom court de l'indicateur, qui est spécifié dans la propriété INDICATOR_SHORTNAMEpar la fonctionIndicatorSetString(). On peut recevoir le nom court de l'indicateur par la fonctionChartIndicatorName(). Pour recevoir l'information surl'erreur, il faut appeler la fonction GetLastError().

Note

Il ne faut pas confondre un nom court et de l'indicateur et le nom du fichier qui est spécifié pendant la création de l'indicateur par les fonctionsiCustom() et IndicatorCreate(). Si le nom court de l'indicateur n'est pas spécifié explicitement, alors pendant la compilation le nom du fichier avec le code initial de l'indicateur y est spécifié.

La suppression de l'indicateur du graphique ne signifie pas que la partie calculé de l'indicateur sera aussi supprimée de la mémoire du terminal. Pour vider le handle de l'indicateur utiliser la fonctionIndicatorRelease().

Il est nécessaire de former correctement un nom court de l'indicateur, qui s'écrit dans la propriété INDICATOR_SHORTNAME à l'aide de la fonctionIndicatorSetString () . Nous recommandons que le nom court contienne les valeurs des paramètres d'entrée de l'indicateur, puisque l'identification de l'indicateur supprimé du graphique dans la fonctionChartIndicatorDelete()est faite selon le nom court.

Voir aussi

ChartIndicatorAdd(), ChartIndicatorDelete(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2024, MetaQuotes Ltd.

ChartIndicatorsTotal

Rend le nombre de tous indicateurs qui sont attachés à la fenêtre indiquée du graphique.

```
int  ChartIndicatorsTotal(
   long  chart_id,       // l'identificateur du graphique
   int   sub_window      // le numéro de la sous-fenêtre
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

La valeur rendue

Le nombre d'indicateurs sur la fenêtre spécifiée du graphique. Pour recevoir l'information sur l'erreur il faut appeler la fonction GetLastError ().

Note

La fonction est conçue pour trier tous les indicateurs attachés à ce graphique. On peut recevoir le nombre de toutes les fenêtres du graphique de la propriété CHART_WINDOWS_TOTAL par la fonction ChartGetInteger().

Voir aussi

ChartIndicatorAdd(), ChartIndicatorDelete(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2024, MetaQuotes Ltd.

ChartWindowOnDropped

Rend le numéro de sous-fenêtre du graphique, sur lequel on jette par la souris l'expert donné, le script, l'objet ou l'indicateur. 0 signifie une fenêtre principale du graphique.

```
int  ChartWindowOnDropped();
```

La valeur rendue

La valeur du type int.

Exemple:

```
   int myWindow=ChartWindowOnDropped();
   int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   Print("Script est lancé sur la fenêtre #"+myWindow+
         ". Au total les fenêtres sur le graphique "+ChartSymbol()+": ",windowsTotal);
```

Voir aussi

ChartPriceOnDropped, ChartTimeOnDropped, ChartXOnDropped, ChartYOnDropped

© 2000-2024, MetaQuotes Ltd.

ChartPriceOnDropped

Rend la coordonnée de prix correspondant au point, dans laquelle on jette par la souris l'expert donné ou le script.

```
double  ChartPriceOnDropped();
```

La valeur rendue

La valeur du type double.

Exemple:

```
   double p=ChartPriceOnDropped();
   Print("ChartPriceOnDropped() = ",p);
```

Voir aussi

ChartXOnDropped, ChartYOnDropped

© 2000-2024, MetaQuotes Ltd.

ChartTimeOnDropped

Rend la coordonnée temporaire correspondant au point, dans laquelle on jette par la souris l'expert donné ou le script.

```
datetime  ChartTimeOnDropped();
```

La valeur rendue

La valeur du type datetime.

Exemple:

```
datetime t=ChartTimeOnDropped();
Print("Script wasdropped on the "+t);
```

Voir aussi

ChartXOnDropped, ChartYOnDropped

© 2000-2024, MetaQuotes Ltd.

ChartXOnDropped

Rend la coordonnée selon l'axe X, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris.

```
int  ChartXOnDropped();
```

La valeur rendue

La valeur de la coordonnée X.

Note

L'axe X est dirigé de gauche à droite.

Exemple:

```
int X=ChartXOnDropped();
int Y=ChartYOnDropped();
Print("(X,Y) = ("+X+","+Y+")");
```

Voir aussi

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2024, MetaQuotes Ltd.

ChartYOnDropped

Rend la coordonnée selon l'axe Y, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris.

```
int  ChartYOnDropped();
```

La valeur rendue

La valeur de la coordonnée Y.

Note

L'axe Y est dirigé de haut en bas.

Voir aussi

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2024, MetaQuotes Ltd.

ChartSetSymbolPeriod

Change le symbole et la période du graphique spécifié. La fonction est asynchrone, c'est à dire qu'elle envoie la commande et n'attend pas la fin de l'exécution. La commande est ajoutée dans la queue des messages du graphique et sera exécutée après l'exécution de toutes les commandes précédentes.

```
bool  ChartSetSymbolPeriod(
   long            chart_id,      // Identifiant du graphique
   string          symbol,        // Nom du symbole
   ENUM_TIMEFRAMES period         // Période
   );
```

Parameters

chart_id

[in] Identifiant du graphique. 0 signifie le graphique courant.

symbol

[in] Symbole du graphique. La valeur NULL signifie le symbole du graphique courant (celui auquel l'Expert Advisor est attaché)

period

[in] Période du graphique. Peut être l'une des valeurs de ENUM_TIMEFRAMES. 0 signifie la période du graphique courant.

Valeur de Retour

Retourne true si la commande a été ajoutée dans la queue du graphique, false sinon. Pour connaître l'erreur, utilisez la fonction GetLastError().

Note

Le changement de symbole/période déclenche la réinitialisation de l'Expert Advisor attaché à un graphique.

L'appel à ChartSetSymbolPeriod avec le même symbole et la même période peut être utilisé pour mettre à jour le graphique (de façon similaire à la commande Rafraîchir du terminal). A son tour, la mise à jour du graphique déclenche le recalcul des indicateurs qui sont atachés. Il est donc possible de calculer un indicateur sur le graphique même s'il n'y a aucun tick (par exemple les weekends).

Voir aussi

ChartSymbol, ChartPeriod

© 2000-2024, MetaQuotes Ltd.

ChartScreenShot

La fonction assure le screenshot du graphique indiqué dans son état courant dans le format GIF, PNG ou BMP en fonction de l'extension spécifiée.

```
bool  ChartScreenShot(
   long            chart_id,                  // identificateur du graphique
   string          filename,                  // nom du fichier
   int             width,                     // largeur
   int             height,                    // hauteur
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT    // type de l'alignement
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

filename

[in] Le nom du fichier de screenshot. Ne peut pas excéder 63 caractères. Le screenshot se place au répertoire \Files.

width

[in] La largeur de screenshot en pixels

height

[in] La hauteur de screenshot en pixels

align_mode=ALIGN_RIGHT

[in] Le mode de la sortie de screenshot étroit. les valeurs de l'énumération ENUM_ALIGN_MODE. ALIGN_RIGHT signifie l'alignement selon la marge droite (la sortie de la fin). ALIGN_LEFT signifie l'alignement selon une marge gauche.

La valeur rendue

Rend true en cas du succès, autrement rend false.

Note

S'il faut de faire un screenshot d'un graphique d'une certaine position, tout d'abord il est nécessaire de positionner le graphique en utilisant la fonction ChartNavigate(). Si la dimension horizontale du screenshot est plus petite que la fenêtre de graphique, la partie droite de la fenêtre du graphique, ou une gauche partie sont affichées, en fonction de la valeur du paramètre align_mode.

Exemple:

```
#property description "Le conseiller présente la création de la série des captures d'é
#property description "à l'aide de la fonction ChartScreenShot(). Le nom du fichier"
#property description "est déduit aussi sur le graphique pour la commodité. La hauteur

#define        WIDTH  800     // la largeur du dessin pour l'appel de ChartScreenShot
#define        HEIGHT 600     // La hauteur du dessin pour l'appel de ChartScreenShot
```

© 2000-2024, MetaQuotes Ltd.

```
//--- input parameters
input int      pictures=5;    // la quantité de dessins dans une série
int            mode=-1;       // -1 signifie le décalage vers le bord droit du graphic
int            bars_shift=300;// le nombre de barres au défilement du graphique par la
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//---désactivons l'autodéfilement du graphique
   ChartSetInteger(0,CHART_AUTOSCROLL,false);
//--- établissons l'alinéa du bord droit du graphique
   ChartSetInteger(0,CHART_SHIFT,true);
//---établissons l'affichage du graphique en forme des bougies
   ChartSetInteger(0,CHART_MODE,CHART_CANDLES);
//---
   Print("La préparation du conseiller au travail est terminée");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- la sortie du nom de la fonction, du temps de l'appel et de l'identificateur de l
   Print(__FUNCTION__,TimeCurrent(),"   id=",id,"   mode=",mode);
//--- le traitement de l'événement CHARTEVENT_CLICK ("La pression par la souris sur le
   if(id==CHARTEVENT_CLICK)
     {
      //--- le déplacement initial du bord du graphique
      int pos=0;
      //--- le mode du travail avec le bord gauche du graphique
      if(mode>0)
        {
         //--- défilons le graphique vers le bord gauche
         ChartNavigate(0,CHART_BEGIN,pos);
         for(int i=0;i<pictures;i++)
           {
            //---préparons la signature sur le graphique et le nom pour le fichier
```

© 2000-2024, MetaQuotes Ltd.

```
            string name="ChartScreenShot"+"CHART_BEGIN"+string(pos)+".gif";
            //--- la conclusion du nom sur le graphique en forme du commentaire
            Comment(name);
            //--- sauvegardons la capture d'écran au dossier le répertoire_du terminal
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_LEFT))
               Print("Nous avons sauvegardé la capture d'écran ",name);
            //---
            pos+=bars_shift;
            //---donnons le temps à l'utilisateur pour regarder un nouveau secteur du
            Sleep(3000);
            //--- défilons le graphique de la position courante au bars_shift à droite
            ChartNavigate(0,CHART_CURRENT_POS,bars_shift);
           }
         //--- le changement du mode à l'opposé
         mode*=-1;
        }
      else // le mode du travail avec le bord droit du graphique
        {
         //--- défilons le graphique vers le bord droit
         ChartNavigate(0,CHART_END,pos);
         for(int i=0;i<pictures;i++)
           {
            //---préparons la signature sur le graphique et le nom pour le fichier
            string name="ChartScreenShot"+"CHART_END"+string(pos)+".gif";
            //--- la conclusion du nom sur le graphique en forme du commentaire
            Comment(name);
            //--- sauvegardons la capture d'écran au dossier le répertoire_du terminal
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_RIGHT))
               Print("Nous avons sauvegardé la capture d'écran ",name);
            //---
            pos+=bars_shift;
            //---donnons le temps à l'utilisateur pour regarder un nouveau secteur du
            Sleep(3000);
            //--- défilons le graphique de la position courante au bars_shift à droite
            ChartNavigate(0,CHART_CURRENT_POS,-bars_shift);
           }
         //--- le changement du mode à l'opposé
         mode*=-1;
        }
     }  // la fin du traitement de l'événement CHARTEVENT_CLICK
//--- la fin du gestionnaire OnChartEvent()
  }
```

Voir aussi

ChartNavigate(), Les ressources

© 2000-2024, MetaQuotes Ltd.

Fonctions de Trading

C'est le groupe de fonctions permettant de gérer les activités de trading.

Avant de commencer à étudier les fonctions de trading de la plateforme, vous devez comprendre les termes de base : ordre, transaction et position.

- Un ordre est une instruction donnée au courtier d'acheter ou de vendre un instrument financier. Il y a deux principaux types d'ordres : au Marché et en Attente. De plus il y a des niveaux spéciaux de Take Profit et de Stop Loss.
- Une transaction est l'échange commercial (achat ou vente) d'un instrument financier. L'Achat est exécuté au prix de la demande (Ask), et la Vente est réalisée au prix de l'offre (Bid). Une transaction peut être ouverte à la suite de l'exécution d'un ordre de marché ou par le déclenchement d'un ordre en attente. Notez que dans certains cas, l'exécution d'un ordre peut donner lieu à plusieures transactions.
- Une position est une obligation commerciale, à savoir le nombre de contrats achetés ou vendus d'un instrument financier. Une position longue est un titre acheté en espérant que le prix de l'instrument va augmenter.

Des informations générales sur les opérations de trading sont disponibles dans l'aide du terminal client.

Les fonctions de trading peuvent être utiliséees dans les Expert Advisors et dans les scripts. Les fonctions de trading ne peuvent être appelées que si l'option "Autoriser le trading live' est activée dans les propriétés de l'Expert Advisor ou du script.

Le trading peut être autorisé ou interdit suivant différents facteurs décrits dans la section Autorisation de Trading.

Fonction	Action
OrderCalcMargin	Calcule la marge requise pour le type d'ordre spécifié, dans la devise du dépôt
OrderCalcProfit	Calcule le profit suivant les paramètres passés, dans la devise du dépôt
OrderCheck	Vérifie s'il y a suffisamment de fonds pour exécuter l'opération de trading demandée.
OrderSend	Envoie des demandes de trade à un serveur
OrderSendAsync	Envoie de façon asynchrone les demandes de trades sans attendre les réponses du serveur de trades
PositionsTotal	Retourne le nombre de positions ouvertes
PositionGetSymbol	Retourne le symbole correspondant à la position ouverte
PositionSelect	Sélectionne une position ouverte pour pouvoir l'utiliser
PositionSelectByTicket	Sélectionne une position par son numéro de ticket pour pouvoir l'utiliser
PositionGetDouble	Retourne la propriété demandée d'une position ouverte (double)

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
PositionGetInteger	Retourne la propriété demandée d'une position ouverte (datetime ou int)
PositionGetString	Retourne la propriété demandée d'une position ouverte (string)
PositionGetTicket	Retourne le ticket de la position à l'indice spécifié dans la liste des positions ouvertes
OrdersTotal	Retourne le nombre d'ordres
OrderGetTicket	Retourne le ticket de l'ordre correspondant
OrderSelect	Sélectionne un ordre pour pouvoir l'utiliser
OrderGetDouble	Retourne la propriété demandée de l'ordre (double)
OrderGetInteger	Retourne la propriété demandée de l'ordre (datetime ou int)
OrderGetString	Retourne la propriété demandée de l'ordre (string)
HistorySelect	Récupère l'historique des transactions et des ordres pour la période spécifiée
HistorySelectByPosition	Récupère l'historique des transactions avec un identifiant de position spécifié.
HistoryOrderSelect	Sélectionne un ordre dans l'historique pour l'utiliser ensuite
HistoryOrdersTotal	Retourne le nombre d'ordres dans l'historique
HistoryOrderGetTicket	Retoure le ticket de l'ordre correspondant dans l'historique
HistoryOrderGetDouble	Retourne la propriété demandée d'un ordre dans l'historique (double)
HistoryOrderGetInteger	Retourne la propriété demandée d'un ordre dans l'historique (datetime ou int)
HistoryOrderGetString	Retourne la propriété demandée d'un ordre dans l'historique (string)
HistoryDealSelect	Sélectionne une transaction dans l'historique pour l'utiliser ensuite avec les fonctions correspondantes
HistoryDealsTotal	Retourne le nombre de transactions dans l'historique
HistoryDealGetTicket	Retourne le ticket de la transaction correspondante dans l'historique
HistoryDealGetDouble	Retourne la propriété demandée d'une transaction dans l'historique (double)
HistoryDealGetInteger	Retourne la propriété demandée d'une transaction dans l'historique (datetime ou int)
HistoryDealGetString	Retourne la propriété demandée d'une transaction dans l'historique (string)

© 2000-2024, MetaQuotes Ltd.

OrderCalcMargin

La fonction calcule la marge requise pour le type d'ordre spécifié, pour le compte courant, dans l'environnement de marché courant, et sans tenir compte des ordres en attente et des positions ouvertes. Elle permet d'évaluer la marge pour l'opération de trading planifiée. La valeur est retournée dans la devise du compte.

```
bool  OrderCalcMargin(
   ENUM_ORDER_TYPE       action,          // type de l'ordre
   string                symbol,          // nom du symbole
   double                volume,          // volume
   double                price,           // prix d'ouverture
   double&               margin           // variable pour récupérer la valeur de la
   );
```

Parameters

action

[in] The order type, can be one of the values of the ENUM_ORDER_TYPE

symbol

[in] Nom du symbole.

volume

[in] Volume de l'opération de trading.

price

[in] Prix d'ouverture.

margin

[out] La variable dans laquelle la valeur de la marge requise sera écrite si la fonction est exécutée avec succès. Le calcul est effectué comme s'il n'y avait pas d'ordre en attente ou de position ouverte sur le compte actuel. La valeur de la marge dépend de plusieurs facteurs et peut différer dans des environnements de marché différents.

Valeur de Retour

Retourne true en cas de succès, false sinon. Pour obtenir des informations sur l'erreur, appelez la fonction GetLastError().

Voir également

OrderSend(), Propriétés d'un Ordre, Types des Opérations de Trading

© 2000-2024, MetaQuotes Ltd.

OrderCalcProfit

La fonction calcule le profit pour le compte courant, dans l'environnement de marché courant et sur la base des paramètres passés. La fonction est utilisée pour pré-évaluer le résultat d'une opération de trading. La valeur est retournée dans la devise du compte.

```
bool  OrderCalcProfit(
   ENUM_ORDER_TYPE       action,           // type de l'ordre (ORDER_TYPE_BUY ou ORDE
   string                symbol,           // nom du symbole
   double                volume,           // volume
   double                price_open,       // prix d'ouverture
   double                price_close,      // prix de clôture
   double&               profit            // variable pour récupérer la valeur du pro
   );
```

Parameters

action

[in] Type de l'ordre, peut être l'une des 2 valeurs de l'énumération ENUM_ORDER_TYPE : ORDER_TYPE_BUY ou ORDER_TYPE_SELL.

symbol

[in] Nom du symbole.

volume

[in] Volume de l'opération de trading.

price_open

[in] Prix d'ouverture.

price_close

[in] Prix de clôture.

profit

[out] La variable dans laquelle la valeur calculée du profit sera écrite si la fonction est exécutée avec succès. La valeur estimée du profit dépend de plusieurs facteurs et peut différer dans des environnements de marché différents.

Valeur de Retour

Retourne true en cas de succès, false sinon. Si un type d'ordre invalide est spécifié, la fonction retourne false. Pour obtenir des informations sur une erreur, appelez GetLastError().

Voir également

OrderSend(), Propriétés d'un Ordre, Types des Opérations de Trading

© 2000-2024, MetaQuotes Ltd.

OrderCheck

La fonction OrderCheck() vérifie s'il y a suffisamment de fonds pour exécuter l'opération de trading demandée. Les résultats de la vérification sont placés dans les champs de la structure MqlTradeCheckResult.

```
bool  OrderCheck(
   MqlTradeRequest&        request,      // structure de la demande
   MqlTradeCheckResult&    result        // structure du résultat
   );
```

Parameters

request

[in] Pointeur vers la structure de type MqlTradeRequest, qui décrit l'action de trading désirée.

result

[in,out] Pointeur vers la structure de type MqlTradeCheckResult, dans laquelle le résultat de la vérification sera placé.

Valeur de Retour

Si les fonds sont insuffisants pour l'opération, ou que les paramètres sont remplis incorrectement, la fonction retourne false. Si la vérification de base des structures réussit (vérification des pointeurs), elle retourne true. **Cependant, ce n'est pas une indication que l'opération de trading désirée sera exécutée avec succès**. Pour une description plus détaillée du résultat de l'exécution de la fonction, analysez les champs de la structure *result* .

Pour obtenir des informations sur l'erreur, appelez la fonction GetLastError().

Voir également

>OrderSend(), Types d'une Opération de Trading, Structure de Demande de Trade, Structure de Résultat de la Vérification d'une Demande, Structure de Résultat d'une Demande de Trade

© 2000-2024, MetaQuotes Ltd.

OrderSend

La fonction OrderSend() est utilisée pour exécuter des opérations de trading en envoyant des demandes à un serveur de trades.

```
bool  OrderSend(
   MqlTradeRequest&  request,      // structure de la demande
   MqlTradeResult&   result        // structure de la réponse
   );
```

Parameters

request

[in] Un pointeur vers une structure de type MqlTradeRequest qui décrit l'activité de trading du client.

result

[in,out] Un pointeur vers une structure de type MqlTradeResult qui décrit le résultat d'une opération de trading en cas d'exécution réussie de la fonction (si true est retourné).

Valeur de Retour

Si la vérification de base des structures réussit (vérification des pointeurs), elle retourne true. **Cependant, ce n'est pas le signe de l'exécution réussie d'une opération de trading**. Pour une description plus détaillée du résultat de l'exécution de la fonction, analysez les champs de la structure *result* .

Note

Les demandes de trades passent par plusieurs étapes de vérification sur le serveur de trades. Premièrement, il vérifie que tous les champs requis des paramètres de la *requête* sont remplis correctement. S'il n'y a pas d'erreur, le serveur accepte l'ordre pour son traitement ultérieur. Si l'ordre est accepté par le serveur de trades, la fonction OrderSend() retourne true.

Il est recommandé de vérifier la requête avant de l'envoyer au serveur de trades. Pour vérifier les demandes, utilisez la fonction OrderCheck(). Il vérifie s'il y a des fonds suffisants pour exécuter l'opération de trading, et retourne différents paramètres utiles dans les résultats de vérification de la demande de trade :

- le code de retour contenant les information sur les erreurs dans la requête vérifiée ;
- le solde qui apparaîtra après l'exécution de l'opération de trading ;
- la valeur des fonds qui apparaîtra après l'exécution de l'opération de trading ;
- la valeur du point qui apparaîtra après l'exécution de l'opération de trading ;
- la marge requise pour l'opération de trading ;
- montant des capitaux libres qui resteront après l'exécution de l'opération de trading ;
- le niveau de marge qui sera déterminé après l'exécution de l'opération de trading ;
- commentaire au code de réponse, descriptionde l'erreur.

Lors de l'envoi d'un ordre au marché (MqlTradeRequest.action=TRADE_ACTION_DEAL), le succès de la fonction OrderSend() ne signifie pas que l'ordre a été exécuté (les trades correspondants ont été effectués). Dans ce cas, 'true' ne signifie que que l'ordre a été placé avec succès dans le système de trading pour son exécution ultérieure. Le serveur de trades peut remplir les *valeurs du champ* de *la transaction ou de l'ordre* dans la structure de résultat retournée, si ces données sont connues au

© 2000-2024, MetaQuotes Ltd.

moment de la formation de la réponse à un appel à OrderSend(). Généralement, les évènements d'exécution des trades correspondant à un ordre peuvent se produire après l'envoi d'une réponse à l'appel OrderSend(). Pour n'importe quel type de demande de trade, lorsque vous recevez le résultat de l'exécution de OrderSend(), vous devriez donc vérifier d'abord le *retcode* du code de réponse du serveur de trades et le code de réponse du système externe *retcode_external* (si nécessaire) disponible dans la structure result obtenue.

Chaque ordre accepté est stocké sur le serveur de trades en attendant sont traitement jusqu'à ce que l'une des conditions pour son exécution se produise :

- expiration,
- arrivée d'une demande opposée,
- exécution de l'ordre lorsque le prix d'exécution apparaît,
- arrivée d'une demande pour annuler l'ordre.

Au moment du traitement de l'ordre, le serveur de trades envoie un message au terminal à propos de l'occurence de l'évènement Trade, qui peut être traité par la fonction OnTrade().

Le résultat de l'exécution de la demande de trade sur un serveur envoyé par la fonction OrderSend() peut être suivi avec la fonction de gestion OnTradeTransaction. Il est à noter que la fonction OnTradeTransaction sera appelée plusieurs fois lors de l'exécution d'une demande de trade.

Par exemple, lors de l'envoi d'un ordre d'achat, il est pris en compte par le gestionnaire, un ordre d'achat correspondant est créé pour le compte, l'ordre est ensuite exécuté et supprimé de la liste des ordres ouverts, puis il est ajouté à l'historique des ordres, une transaction correspondante est ajoutée à l'hisorique et une nouvelle position est créée. La fonction OnTradeTransaction sera appelée pour chacun de ces évènements.

Exemple :

```
//--- valeur de ORDER_MAGIC
input long order_magic=55555;
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- on s'assure que c'est un compte de démo
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("L'utilisation d'un script n'est pas autorisée sur un compte réel !");
      return;
     }
//--- place ou suprimer des ordres
   if(GetOrdersTotalByMagic(order_magic)==0)
     {
      //--- aucun ordre en cours, place un ordre
      uint res=SendRandomPendingOrder(order_magic);
      Print("Code de retour du serveur de trades ",res);
     }
   else // il y a des ordres, supprime les ordres
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      DeleteAllOrdersByMagic(order_magic);
     }
//---
  }
//+------------------------------------------------------------------+
//| Retourne le nombre d'ordres en cours avec l'ORDER_MAGIC spécifié |
//+------------------------------------------------------------------+
int GetOrdersTotalByMagic(long const magic_number)
  {
   ulong order_ticket;
   int total=0;
//--- parcours tous les ordres en attente
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
         if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
//---
   return(total);
  }
//+------------------------------------------------------------------+
//| Supprime tous les ordres en attente avec l'ORDER_MAGIC spécifié  |
//+------------------------------------------------------------------+
void DeleteAllOrdersByMagic(long const magic_number)
  {
   ulong order_ticket;
//--- parcours tous les ordres en attente
   for(int i=OrdersTotal()-1;i>=0;i--)
      if((order_ticket=OrderGetTicket(i))>0)
         //--- ordre avec l'ORDER_MAGIC spécifié
         if(magic_number==OrderGetInteger(ORDER_MAGIC))
           {
            MqlTradeResult result={};
            MqlTradeRequest request={};
            request.order=order_ticket;
            request.action=TRADE_ACTION_REMOVE;
            OrderSend(request,result);
            //--- écrit la réponse du serveur dans le journal
            Print(__FUNCTION__,": ",result.comment," code de réponse ",result.retcode)
           }
//---
  }
//+------------------------------------------------------------------+
//| Définit un ordre en attente de façon aléatoire                   |
//+------------------------------------------------------------------+
uint SendRandomPendingOrder(long const magic_number)
  {
//--- prépare une requête
   MqlTradeRequest request={};
   request.action=TRADE_ACTION_PENDING;         // définit un ordre en attente
   request.magic=magic_number;                  // ORDER_MAGIC
```

© 2000-2024, MetaQuotes Ltd.

```
   request.symbol=_Symbol;                          // symbole
   request.volume=0.1;                              // volume de 0,1 lots
   request.sl=0;                                    // le Stop Loss n'est pas spécifié
   request.tp=0;                                    // le Take Profit n'est pas spécifié
//--- définit le type de l'ordre
   request.type=GetRandomType();                    // type de l'ordre
//--- définit le prix de l'ordre en attente
   request.price=GetRandomPrice(request.type);  // prix d'ouverture
//--- envoie la demande de trade
   MqlTradeResult result={};
   OrderSend(request,result);
//--- écrit la réponse du serveur dans le journal
   Print(__FUNCTION__," :",result.comment);
   if(result.retcode==10016) Print(result.bid,result.ask,result.price);
//--- code de retour de la réponse du serveur de trade
   return result.retcode;
  }
//+------------------------------------------------------------------+
//| Retourne un type d'ordre en attente de façon aléatoire           |
//+------------------------------------------------------------------+
ENUM_ORDER_TYPE GetRandomType()
  {
   int t=MathRand()%4;
//---   0<=t<4
   switch(t)
     {
      case(0):return(ORDER_TYPE_BUY_LIMIT);
      case(1):return(ORDER_TYPE_SELL_LIMIT);
      case(2):return(ORDER_TYPE_BUY_STOP);
      case(3):return(ORDER_TYPE_SELL_STOP);
     }
//--- valeur incorrecte
   return(WRONG_VALUE);
  }
//+------------------------------------------------------------------+
//| Retourne un prix de façon aléatoire                              |
//+------------------------------------------------------------------+
double GetRandomPrice(ENUM_ORDER_TYPE type)
  {
   int t=(int)type;
//--- niveaux de stops pour le symbole
   int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- récupère les données du dernier tick
   MqlTick last_tick={};
   SymbolInfoTick(_Symbol,last_tick);
//--- calcule le prix selon le type
   double price;
   if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT ou ORDER_TYPE_SELL_STOP
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      price=last_tick.bid; // à partir du prix Bid
      price=price-(distance+(MathRand()%10)*5)*_Point;
     }
   else              // ORDER_TYPE_SELL_LIMIT ou ORDER_TYPE_BUY_STOP
     {
      price=last_tick.ask; // à partir du price Ask
      price=price+(distance+(MathRand()%10)*5)*_Point;
     }
//---
   return(price);
  }
```

Voir également

Types d'une Opération de Trading, Structure de Demande de Trade, Structure de Résultat de la Vérification d'une Demande, Structure de Résultat d'une Demande de Trade

© 2000-2024, MetaQuotes Ltd.

OrderSendAsync

La fonction OrderSendAsync() est utilisée pour effectuer des opérations de trading de façon asynchrone sans attendre une réponse du serveur de trades à une demande envoyée. La fonction est conçue pour le trading à haute fréquence, quand selon les termes de l'algorithme de trading il n'est pas acceptable d'attendre une réponse du serveur.

```
bool  OrderSendAsync(
   MqlTradeRequest&  request,      // structure de la demande
   MqlTradeResult&   result        // structure de la réponse
   );
```

Parameters

request

[in] Un pointeur vers une structure de type MqlTradeRequest qui décrit l'action de trading du client.

result

[in,out] Un pointeur vers une structure de type MqlTradeResult qui décrit le résultat d'une opération de trading en cas d'exécution réussie de la fonction (si true est retourné).

Valeur de Retour

Retourne true si la demande est envoyée au serveur de trades. Si la demande n'est pas envoyée, retourne false. Si la demande est envoyée, le code de réponse dans la variable *result* contient la valeur TRADE_RETCODE_PLACED (code 10008) - "ordre placé". Une exécution réussie ne signifie que l'envoi, mais ne donne aucune garantie que la demande a atteint le serveur de trades et a été acceptée. Lors du traitement de la demande reçue, un serveur de trades envoie une réponse au terminal client notifiant un changement dans l'état actuel des positions, des ordres et des transactions, qui amène à la génération de l'évènement Trade.

Le résultat de l'exécution de la demande de trade sur un serveur envoyée par la fonction OrderSendAsync() peut être suivi avec la fonction OnTradeTransaction. Il est à noter que la fonction OnTradeTransaction sera appelée plusieurs fois lors de l'exécution d'une demande de trade.

Par exemple, lors de l'envoi d'un ordre d'achat, il est pris en compte par le gestionnaire, un ordre d'achat correspondant est créé pour le compte, l'ordre est ensuite exécuté et supprimé de la liste des ordres ouverts, puis il est ajouté à l'historique des ordres, une transaction correspondante est ajoutée à l'hisorique et une nouvelle position est créée. La fonction OnTradeTransaction sera appelée pour chacun de ces évènements. Pour obtenir ces données, les paramètres de la fonction doivent être analysés :

- **trans** - ce paramètre prend une structure MqlTradeTransaction décrivant une transaction de trading appliquée à un compte de trading ;
- **request** - ce paramètre prend une structure MqlTradeRequest décrivant la demande de trade ayant résulté en une transaction de trade ;
- **result** - ce paramètre prend une structure MqlTradeResult décrivant le résultat de l'exécution de la demande de trade.

Note

En termes d'objectifs et de paramètres, la fonction est similaire à OrderSend(), mais contrairement à elle, elle est asynchrone, c'est à dire qu'elle ne bloque pas l'exécution du programme en attendant

© 2000-2024, MetaQuotes Ltd.

le résultat de l'exécution de la fonction. Vous pouvez comparer le taux d'opérations de trades de ces 2 fonctions en utilisant l'Expert Advisor exemple.

Exemple :

```
#property description "Expert Advisor qui envoie des demandes de trades "
                      " en utilisant la fonction OrderSendAsync().\r\n"
#property description "La gestion des évènements de trading gérés avec"
                      " les fonctions OnTrade() et OnTradeTransaction() est affichée\r
#property description "Les paramètres de l'Expert Advisor permet de définir le Magic N
                      " (identifiant unique) "
#property description "et le mode d'affichage des messages dans le journal des Experts
//--- paramètres d'entrée
input int  MagicNumber=1234567;      // Identifiant de l'Expert Advisor
input bool DescriptionModeFull=true; // Mode de sortie détaillée
//--- variable à utiliser dans l'appel à HistorySelect()
datetime history_start;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vérifie si le trading automatique est autorisé
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
     {
      Alert("Le trading automatique dans le terminal est désactivé, l'Expert Advisor s
      ExpertRemove();
      return(-1);
     }
//--- impossible de trader sur un compte réel
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("L'Expert Advisor ne peut pas trader sur un compte réel !");
      ExpertRemove();
      return(-2);
     }
//--- vérifie s'il est possible de trader sur ce compte (par exemple, le trading est 
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
     {
      Alert("Le trading sur ce compte est désactivé");
      ExpertRemove();
      return(-3);
     }
//--- sauvegarde l'heure de lancement de l'Expert Advisor pour récupérer l'historique
   history_start=TimeCurrent();
//---
   CreateBuySellButtons();
   return(INIT_SUCCEEDED);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Fonction de dé-initialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprime tous les objets graphiques
   ObjectDelete(0,"Buy");
   ObjectDelete(0,"Sell");
//---
  }
//+------------------------------------------------------------------+
//| Fonction TradeTransaction                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- en-tête nommé d'après la fonction de gestion de l'évènement de trading
   Print("=> ",__FUNCTION__," à ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- récupère le type de transaction sous la forme d'une valeur d'énumération
   ENUM_TRADE_TRANSACTION_TYPE type=trans.type;
//--- si la transaction est le réstulat de la gestion de la demande
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      //--- affiche le nom de la transaction
      Print(EnumToString(type));
      //--- affiche ensuite la description de la requête traitée
      Print("------------RequestDescription\r\n",
            RequestDescription(request,DescriptionModeFull));
      //--- et affiche la description du résultat de la requête
      Print("------------ ResultDescription\r\n",
            TradeResultDescription(result,DescriptionModeFull));
     }
   else // affiche la description complète de la transaction pour les transactions d'u
     {
      Print("------------ TransactionDescription\r\n",
            TransactionDescription(trans,DescriptionModeFull));
     }
//---
  }
//+------------------------------------------------------------------+
//| Fonction Trade                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- membres statiques pour stocket l'état du compte de trading
   static int prev_positions=0,prev_orders=0,prev_deals=0,prev_history_orders=0;
//--- récupère l'historique de trading
   bool update=HistorySelect(history_start,TimeCurrent());
```

© 2000-2024, MetaQuotes Ltd.

```
   PrintFormat("HistorySelect(%s , %s) = %s",
               TimeToString(history_start),TimeToString(TimeCurrent()),(string)update)
//--- en-tête nommé d'après la fonction de gestion de l'évènement de trading
   Print("=> ",__FUNCTION__," à ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- affiche le nom de la fonction et le nombre d'ordre au moment de l'appel
   int curr_positions=PositionsTotal();
   int curr_orders=OrdersTotal();
   int curr_deals=HistoryOrdersTotal();
   int curr_history_orders=HistoryDealsTotal();
//--- affiche le nombre d'ordres, de positions, de transactions ainsi que les changeme
   PrintFormat("PositionsTotal() = %d (%+d)",
               curr_positions,(curr_positions-prev_positions));
   PrintFormat("OrdersTotal() = %d (%+d)",
               curr_orders,curr_orders-prev_orders);
   PrintFormat("HistoryOrdersTotal() = %d (%+d)",
               curr_deals,curr_deals-prev_deals);
   PrintFormat("HistoryDealsTotal() = %d (%+d)",
               curr_history_orders,curr_history_orders-prev_history_orders);
//--- insère une séparation pour aider à la lecture du journal
   Print("");
//--- sauvegarde le statut du compte
   prev_positions=curr_positions;
   prev_orders=curr_orders;
   prev_deals=curr_deals;
   prev_history_orders=curr_history_orders;
//---
  }
//+------------------------------------------------------------------+
//| Fonction ChartEvent                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- gère l'évènement CHARTEVENT_CLICK ("Clic sur le graphique")
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("=> ",__FUNCTION__,": sparam = ",sparam);
      //--- volume minimum pour une transaction
      double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
      //--- si le bouton "Buy" est utilisé, alors achat
      if(sparam=="Buy")
        {
         PrintFormat("Buy %s %G lot",_Symbol,volume_min);
         BuyAsync(volume_min);
         //--- relâche le bouton
         ObjectSetInteger(0,"Buy",OBJPROP_STATE,false);
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- si le bouton "Sell" est utilisé, alors vente
      if(sparam=="Sell")
        {
         PrintFormat("Sell %s %G lot",_Symbol,volume_min);
         SellAsync(volume_min);
         //--- relâche le bouton
         ObjectSetInteger(0,"Sell",OBJPROP_STATE,false);
        }
      ChartRedraw();
     }
//---
  }
//+------------------------------------------------------------------+
//| Retourne la description textuelle d'une transaction              |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans,
                              const bool detailed=true)
  {
//--- prépare une chaîne de caractère pour le retour de la fonction
   string desc=EnumToString(trans.type)+"\r\n";
//--- toutes les données possibles sont ajoutées dans le mode détaillé
   if(detailed)
     {
      desc+="Symbole : "+trans.symbol+"\r\n";
      desc+="Ticket de la transaction : "+(string)trans.deal+"\r\n";
      desc+="Type de la transaction : "+EnumToString(trans.deal_type)+"\r\n";
      desc+="Ticket de l'ordre : "+(string)trans.order+"\r\n";
      desc+="Type de l'ordre : "+EnumToString(trans.order_type)+"\r\n";
      desc+="Etat de l'ordre : "+EnumToString(trans.order_state)+"\r\n";
      desc+="Type d'heure de l'ordre : "+EnumToString(trans.time_type)+"\r\n";
      desc+="Expiration de l'ordre : "+TimeToString(trans.time_expiration)+"\r\n";
      desc+="Prix : "+StringFormat("%G",trans.price)+"\r\n";
      desc+="Prix de déclenchement : "+StringFormat("%G",trans.price_trigger)+"\r\n";
      desc+="Stop Loss : "+StringFormat("%G",trans.price_sl)+"\r\n";
      desc+="Take Profit : "+StringFormat("%G",trans.price_tp)+"\r\n";
      desc+="Volume : "+StringFormat("%G",trans.volume)+"\r\n";
     }
//--- retourne la chaîne de caractères
   return desc;
  }
//+------------------------------------------------------------------+
//| Retourne la description textuelle de la demande de trade         |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request,
                          const bool detailed=true)
  {
//--- prépare une chaîne de caractère pour le retour de la fonction
   string desc=EnumToString(request.action)+"\r\n";
//--- ajoute toutes les données disponibles en mode détaillé
```

```
   if(detailed)
     {
      desc+="Symbole : "+request.symbol+"\r\n";
      desc+="Magic Number : "+StringFormat("%d",request.magic)+"\r\n";
      desc+="Ticket de l'ordre : "+(string)request.order+"\r\n";
      desc+="Type de l'ordre : "+EnumToString(request.type)+"\r\n";
      desc+="Remplissage de l'ordre : "+EnumToString(request.type_filling)+"\r\n";
      desc+="Type d'heure de l'ordre : "+EnumToString(request.type_time)+"\r\n";
      desc+="Expiration de l'ordre : "+TimeToString(request.expiration)+"\r\n";
      desc+="Prix : "+StringFormat("%G",request.price)+"\r\n";
      desc+="Points de déviation : "+StringFormat("%G",request.deviation)+"\r\n";
      desc+="Stop Loss : "+StringFormat("%G",request.sl)+"\r\n";
      desc+="Take Profit : "+StringFormat("%G",request.tp)+"\r\n";
      desc+="Stop Limit : "+StringFormat("%G",request.stoplimit)+"\r\n";
      desc+="Volume : "+StringFormat("%G",request.volume)+"\r\n";
      desc+="Commentaire : "+request.comment+"\r\n";
     }
//--- retourne la chaîne de caractères
   return desc;
  }
//+------------------------------------------------------------------+
//| Retourne la description du résultat du traitement de la demande  |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result,
                              const bool detailed=true)
  {
//--- prépare une chaîne de caractère pour le retour de la fonction
   string desc="Retcode "+(string)result.retcode+"\r\n";
//--- ajoute toutes les données disponibles en mode détaillé
   if(detailed)
     {
      desc+="Identifiant de la demande : "+StringFormat("%d",result.request_id)+"\r\n"
      desc+="Ticket de l'ordre : "+(string)result.order+"\r\n";
      desc+="Ticket de la transaction : "+(string)result.deal+"\r\n";
      desc+="Volume : "+StringFormat("%G",result.volume)+"\r\n";
      desc+="Prix : "+StringFormat("%G",result.price)+"\r\n";
      desc+="Ask : "+StringFormat("%G",result.ask)+"\r\n";
      desc+="Bid : "+StringFormat("%G",result.bid)+"\r\n";
      desc+="Commentaire : "+result.comment+"\r\n";
     }
//--- retourne la chaîne de caractères
   return desc;
  }
//+------------------------------------------------------------------+
//| Crée 2 boutons pour acheter et vendre                            |
//+------------------------------------------------------------------+
void CreateBuySellButtons()
  {
//--- vérifie l'objet nommé "Buy"
```

© 2000-2024, MetaQuotes Ltd.

```
   if(ObjectFind(0,"Buy")>=0)
     {
      //--- si l'objet trouvé n'est pas un bouton, le supprime
      if(ObjectGetInteger(0,"Buy",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Buy");
     }
   else
      ObjectCreate(0,"Buy",OBJ_BUTTON,0,0,0); // crée le bouton "Buy"
//--- configure le bouton "Buy"
   ObjectSetInteger(0,"Buy",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Buy",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Buy",OBJPROP_YDISTANCE,50);
   ObjectSetInteger(0,"Buy",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Buy",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Buy",OBJPROP_TEXT,"Buy");
   ObjectSetInteger(0,"Buy",OBJPROP_COLOR,clrRed);
//--- vérifie la présence de l'objet nommé "Sell"
   if(ObjectFind(0,"Sell")>=0)
     {
      //--- si l'objet trouvé n'est pas un bouton, le supprime
      if(ObjectGetInteger(0,"Sell",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Sell");
     }
   else
      ObjectCreate(0,"Sell",OBJ_BUTTON,0,0,0); // crée le bouton "Sell"
//--- configure le bouton "Sell"
   ObjectSetInteger(0,"Sell",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Sell",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Sell",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Sell",OBJPROP_TEXT,"Sell");
   ObjectSetInteger(0,"Sell",OBJPROP_COLOR,clrBlue);
//--- force la mise à jour du graphique pour afficher les boutons immédiatement
   ChartRedraw();
//---
  }
//+------------------------------------------------------------------+
//| Achat avec la fonction asynchrone OrderSendAsync()               |
//+------------------------------------------------------------------+
void BuyAsync(double volume)
  {
//--- prépare la requête
   MqlTradeRequest req={};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_BUY;
```

© 2000-2024, MetaQuotes Ltd.

```
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_ASK);
   req.deviation   =10;
   req.comment     ="Achat avec OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__," : erreur ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
//| Vente avec la fonction asynchrone OrderSendAsync()               |
//+------------------------------------------------------------------+
void SellAsync(double volume)
  {
//--- prépare la requête
   MqlTradeRequest req={};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_SELL;
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_BID);
   req.deviation   =10;
   req.comment     ="Vente avec OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__," : erreur ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
```

Exemple des messages dans le journal "Experts" :

```
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnChartEvent: sparam = Sell
12:52:52   ExpertAdvisor (EURUSD,H1)   Sell EURUSD 0.01 lot
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction à 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_REQUEST
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------RequestDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_ACTION_DEAL
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbole : EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Magic Number : 1234567
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de l'ordre : 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Type de l'ordre : ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Remplissage de l'ordre : ORDER_FILLING_FOK
12:52:52   ExpertAdvisor (EURUSD,H1)   Type d'heure de l'ordre : ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Expiration de l'ordre : 1970.01.01 00:00
```

© 2000-2024, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Prix : 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Points de déviation : 10
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Limit : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume : 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Commentaire : Vente avec OrderSendAsync()
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ ResultDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   Retcode 10009
12:52:52   ExpertAdvisor (EURUSD,H1)   Identifiant de la requête : 2
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de l'ordre : 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de la transaction : 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume : 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Prix : 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Ask : 1.29319
12:52:52   ExpertAdvisor (EURUSD,H1)   Bid : 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Commentaire :
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade à 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+1)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction à 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbole : EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de la transaction : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Type de la transaction : DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de l'ordre : 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Type de l'ordre : ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Etat de l'ordre : ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Type d'heure de l'ordre : ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Expiration de l'ordre : 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Prix : 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Prix de déclenchement : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume : 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction à 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_DELETE
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbole : EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de la transaction : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Type de la transaction : DEAL_TYPE_BUY
```

© 2000-2024, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de l'ordre : 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Type de l'ordre : ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Etat de l'ordre : ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Type d'heure de l'ordre : ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Expiration de l'ordre : 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Prix : 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Prix de déclenchement : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume : 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade à 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction à 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_HISTORY_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbole : EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de la transaction : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Type de la transaction : DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de l'ordre : 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Type de l'ordre : ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Etat de l'ordre : ORDER_STATE_FILLED
12:52:52   ExpertAdvisor (EURUSD,H1)   Type d'heure de l'ordre : ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Expiration de l'ordre : 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Prix : 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Prix de déclenchement : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume : 0
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade à 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction à 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_DEAL_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbole : EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de la transaction : 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Type de la transaction : DEAL_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket de l'ordre : 16361998
```

© 2000-2024, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Type de l'ordre : ORDER_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Etat de l'ordre : ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Type d'heure de l'ordre : ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Expiration de l'ordre : 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Prix : 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Prix de déclenchement : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit : 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume : 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade à 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
```

© 2000-2024, MetaQuotes Ltd.

PositionsTotal

Retourne le nombre de positions ouvertes.

```
int  PositionsTotal();
```

Valeur de Retour

Valeur du type int.

Note

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

Voir également

PositionGetSymbol(), PositionSelect(), Propriétés d'une Position

© 2000-2024, MetaQuotes Ltd.

PositionGetSymbol

Retourne le symbole correspondant à la position ouverte et sélectionne automatiquement la position pour l'utiliser ensuite avec les fonctions PositionGetDouble, PositionGetInteger, PositionGetString.

```
string  PositionGetSymbol(
   int  index      // indice dans la liste des positions
   );
```

Parameters

index

[in] Numéro de la position dans la liste des positions ouvertes.

Valeur de Retour

Valeur de type string. Si la position n'est pas trouvée, retourne une chaîne vide. Pour connaître le code d'erreur, utilisez la fonction GetLastError().

Note

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

Voir également

PositionsTotal(), PositionSelect(), Propriétés d'une Position

© 2000-2024, MetaQuotes Ltd.

PositionSelect

Sélectionne une position ouverte pour pouvoir l'utiliser. Retourne si la fonction s'est terminé avec succès. Retourne false en cas d'échec. Pour plus de détails sur l'erreur, appelez GetLastError().

```
bool  PositionSelect(
   string  symbol      // nom du symbole
   );
```

Parameters

symbol

[in] Nom de l'instrument financier.

Valeur de Retour

Valeur de type bool.

Note

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole. Dans ce cas, PositionSelect sélectionnera la position avec le ticket le plus petit.

La fonction PositionSelect() copie les données d'une position dans l'environnement du programme et les appels ultérieurs à PositionGetDouble(), PositionGetInteger() et PositionGetString() retournent les données copiées auparavant. Cela signifie que la position elle-même peut ne plus exister (ou que son volume, sa direction, etc. a changé), mais que les données de cette position sont toujours accessibles. Pour être sûr d'utiliser les données à jour d'une position, il est recommandé d'appeler PositionSelect() juste avant son utilisation.

Voir également

PositionGetSymbol(), PositionsTotal(), Propriétés d'une Position

© 2000-2024, MetaQuotes Ltd.

PositionSelectByTicket

Sélectionne une position ouverte par son numéro de ticket spécifié pour pouvoir l'utiliser. Retourne true en cas de succès. Sinon retourne false. Appelez GetLastError() pour obtenir des détails sur l'erreur.

```
bool  PositionSelectByTicket(
   ulong   ticket     // Ticket de la position
   );
```

Parameters

ticket

[in] Ticket de la position.

Valeur de Retour

Une valeur de type bool.

Note

La fonction PositionSelectByTicket() copie les données de la position dans l'environnement du programme. Les appels ultérieurs à PositionGetDouble(), PositionGetInteger() et PositionGetString() retournent les données précédemment copiées. Même si une position n'existe plus (ou que son volume, sa direction, etc. a changé), les données sont toujours disponibles. Pour être sûr de recevoir des données valides de la position, il est recommandé d'appeler PositionSelectByTicket() juste avant son utilisation.

Voir également

PositionGetSymbol(), PositionsTotal(), Propriétés d'une Position

© 2000-2024, MetaQuotes Ltd.

PositionGetDouble

La fonction retourne la propriété demandée d'une position ouverte, pré-sélectionnée avec PositionGetSymbol ou PositionSelect. La propriété de la position doit être de type double. Il existe 2 variantes de cette fonction.

1. Retourne immédiatement la valeur de la propriété.

```
double  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id      // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référence (le dernier paramètre).

```
bool  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id,      // Identifiant de la propriété
   double&                        double_var        // Valeur de la propriété
   );
```

Parameters

property_id

[in] Identifiant de la propriété de la position. La valeur peut être l'une des valeurs de l'énumération ENUM_POSITION_PROPERTY_DOUBLE

double_var

[out] Variable de type double qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type double. Si la fonction échoue, retourne 0.

Note

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

Pour être sûr d'utiliser les données à jour d'une position, il est recommandé d'appeler PositionSelect() juste avant son utilisation.

Voir également

PositionGetSymbol(), PositionSelect(), Propriétés d'une Position

© 2000-2024, MetaQuotes Ltd.

PositionGetInteger

La fonction retourne la propriété demandée d'une position ouverte, pré-sélectionnée avec PositionGetSymbol ou PositionSelect. La propriété de la position doit être de type datetime ou int. Il existe 2 variantes de cette fonction.

1. Retourne immédiatement la valeur de la propriété.

```
long  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id      // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référence (le dernier paramètre).

```
bool  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id,     // Identifiant de la propriété
   long&                           long_var         // Valeur de la propriété
   );
```

Parameters

property_id

[in] Identifiant de la propriété de la position. La valeur peut être l'une des valeurs de l'énumération ENUM_POSITION_PROPERTY_INTEGER

long_var

[out] Variable de type long qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type long. Si la fonction échoue, retourne 0.

Note

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

Pour être sûr d'utiliser les données à jour d'une position, il est recommandé d'appeler PositionSelect() juste avant son utilisation.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction Trade                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- vérifie si la position est présente et affiche l'heure de son changement
   if(PositionSelect(_Symbol))
     {
//--- récupère l'identifiant de la position pour pouvoir l'utiliser ensuite
      ulong position_ID=PositionGetInteger(POSITION_IDENTIFIER);
      Print(_Symbol," position #",position_ID);
//--- récupère l'heure de création de la position en millisecondes depuis le 01.01.197
      long create_time_msc=PositionGetInteger(POSITION_TIME_MSC);
      PrintFormat("Position #%d  POSITION_TIME_MSC = %i64 millisecondes => %s",positio
                  create_time_msc,TimeToString(create_time_msc/1000));
//--- récupère l'heure du dernier changement de la position en secondes depuis le 01.0
      long update_time_sec=PositionGetInteger(POSITION_TIME_UPDATE);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE = %i64 secondes => %s",
                  position_ID,update_time_sec,TimeToString(update_time_sec));
//--- récupère l'heure du dernier changement de la position en millisecondes depuis le
      long update_time_msc=PositionGetInteger(POSITION_TIME_UPDATE_MSC);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE_MSC = %i64 millisecondes => %s",
                  position_ID,update_time_msc,TimeToString(update_time_msc/1000));
     }
//---
  }
```

Voir également

PositionGetSymbol(), PositionSelect(), Propriétés d'une Position

© 2000-2024, MetaQuotes Ltd.

PositionGetString

La fonction retourne la propriété demandée d'une position ouverte, pré-sélectionnée avec PositionGetSymbol ou PositionSelect. La propriété de la position doit être de type string. Il existe 2 variantes de cette fonction.

1. Retourne immédiatement la valeur de la propriété.

```
string  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id      // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référence (le dernier paramètre).

```
bool  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id,     // Identifiant de la propriété
   string&                        string_var       // Valeur de la propriété
   );
```

Parameters

property_id

[in] Identifiant de la propriété de la position. La valeur peut être l'une des valeurs de l'énumération ENUM_POSITION_PROPERTY_STRING

string_var

[out] Variable de type string qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type string. Si la fonction échoue, retourne une chaîne vide.

Note

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

Pour être sûr d'utiliser les données à jour d'une position, il est recommandé d'appeler PositionSelect() juste avant son utilisation.

Voir également

PositionGetSymbol(), PositionSelect(), Propriétés d'une Position

© 2000-2024, MetaQuotes Ltd.

PositionGetTicket

La fonction retourne le ticket de la position située à l'indice spécifié dans la liste des positions ouvertes et sélectionne automatiquement la position pour l'utiliser ensuite avec les fonctions PositionGetDouble, PositionGetInteger, PositionGetString.

```
ulong  PositionGetTicket(
   int  index      // Le numéro de la position dans la liste
   );
```

Parameters

index

[in] L'indice d'une position dans la liste des positions ouvertes, la numérotation commence à 0.

Valeur de Retour

Le ticket de la position. Retourne 0 si la fonction échoue.

Note

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

Pour être sûr d'utiliser les données à jour d'une position, il est recommandé d'appeler PositionSelect() juste avant son utilisation.

Voir également

PositionGetSymbol(), PositionSelect(), Propriétés d'une Position

© 2000-2024, MetaQuotes Ltd.

OrdersTotal

Retourne le nombre d'ordres en cours.

```
int  OrdersTotal();
```

Valeur de Retour

Valeur de type int.

Note

Ne confondez pas les ordres en attente en cours avec les positions, qui sont également affichées dans l'onglet Trading de la fenêtre Boîte à Outils. Un ordre est une demande (ou requête) d'effectuer une transaction, tandis qu'une position est le résultat de l'une ou plusieurs transactions.

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

Voir également

OrderSelect(), OrderGetTicket(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

OrderGetTicket

Retourne le ticket de l'ordre correspondant et sélectionne automatiquement l'ordre pour pouvoir l'utiliser ensuite avec les fonctions.

```
ulong  OrderGetTicket(
   int  index      // indice dans la liste des ordres
   );
```

Parameters

index

[in] Numéro de l'ordre dans la liste des ordres en cours.

Valeur de Retour

Valeur de type ulong. Si la fonction échoue, retourne 0.

Note

Ne confondez pas les ordres en attente en cours avec les positions, qui sont également affichées dans l'onglet Trading de la fenêtre Boîte à Outils. Un ordre est une demande (ou requête) d'effectuer une transaction, tandis qu'une position est le résultat de l'une ou plusieurs transactions.

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

La fonction OrderGetTicket() copie les données d'une position dans l'environnement du programme et les appels ultérieurs à OrderGetDouble(), OrderGetInteger() et OrderGetString() retournent les données copiées auparavant. Cela signifie que l'ordre lui-même peut ne plus exister (ou que son prix d'ouverture, son Stop Loss, son Take Profit ou son expiration a changé), mais que les données de cet ordres sont toujours accessibles. Pour être sûr d'utiliser les données à jour d'un ordre, il est recommandé d'appeler OrderGetTicket() juste avant son utilisation.

Exemple :

```
void OnStart()
  {
//--- variables pour retourner les valeurs des propriétés de l'ordre
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- nombre d'ordres en attente en cours
```

© 2000-2024, MetaQuotes Ltd.

```
   uint     total=OrdersTotal();
//--- parcours les ordres dans une boucle
   for(uint i=0;i<total;i++)
     {
      //--- retourne le ticket de l'ordre par sa position dans la liste
      if((ticket=OrderGetTicket(i))>0)
        {
         //--- retourne les propriétés de l'ordre
         open_price    =OrderGetDouble(ORDER_PRICE_OPEN);
         time_setup    =(datetime)OrderGetInteger(ORDER_TIME_SETUP);
         symbol        =OrderGetString(ORDER_SYMBOL);
         order_magic   =OrderGetInteger(ORDER_MAGIC);
         positionID    =OrderGetInteger(ORDER_POSITION_ID);
         initial_volume=OrderGetDouble(ORDER_VOLUME_INITIAL);
         type          =EnumToString(ENUM_ORDER_TYPE(OrderGetInteger(ORDER_TYPE)));
         //--- prépare et affiche les informations de l'ordre
         printf("#ticket %d %s %G %s à %G a été établie à %s",
                ticket,                 // ticket de l'ordre
                type,                   // type
                initial_volume,         // volume placé
                symbol,                 // symbole
                open_price,             // prix d'ouverture spécifié
                TimeToString(time_setup)// heure du placement de l'ordre
                );
        }
     }
//---
  }
```

Voir également

OrdersTotal(), OrderSelect(), OrderGetInteger()

© 2000-2024, MetaQuotes Ltd.

OrderSelect

Sélectionne un ordre pour pouvoir l'utiliser. Retourne si la fonction s'est terminé avec succès. Sinon retourne false. Pour plus de détails sur l'erreur, appelez GetLastError().

```
bool  OrderSelect(
   ulong   ticket      // Ticket de l'ordre
   );
```

Parameters

ticket

[in] Ticket de l'ordre.

Valeur de Retour

Valeur de type bool.

Note

Ne confondez pas les ordres en attente en cours avec les positions, qui sont également affichées dans l'onglet Trading de la fenêtre Boîte à Outils.

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

La fonction OrderSelect() copie les données d'une position dans l'environnement du programme et les appels ultérieurs à OrderGetDouble(), OrderGetInteger() et OrderGetString() retournent les données copiées auparavant. Cela signifie que l'ordre lui-même peut ne plus exister (ou que son prix d'ouverture, son Stop Loss, son Take Profit ou son expiration a changé), mais que les données de cet ordres sont toujours accessibles. Pour être sûr d'utiliser les données à jour d'un ordre, il est recommandé d'appeler OrderSelect() juste avant son utilisation.

Voir également

OrderGetInteger(), OrderGetDouble(), OrderGetString(), OrderCalcProfit(), OrderGetTicket(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

OrderGetDouble

Retourne la propriété demandée d'un ordre, pré-sélectionné avec OrderGetTicket ou OrderSelect. La propriété de l'ordre doit être de type double. Il existe 2 variantes de cette fonction.

1. Retourne immédiatement la valeur de la propriété.

```
double  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id          // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référénce (le dernier paramètre).

```
bool  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,        // Identifiant de la propriété
   double&                     double_var          // Valeur de la propriété
   );
```

Parameters

property_id

[in] Identifiant de la propriété de l'ordre. La valeur peut être l'une des valeurs de l'énumération ENUM_ORDER_PROPERTY_DOUBLE

double_var

[out] Variable de type double qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type double. Si la fonction échoue, retourne 0.

Note

Ne confondez pas les ordres en attente en cours avec les positions, qui sont également affichées dans l'onglet Trading de la fenêtre Boîte à Outils.

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

Pour être sûr d'utiliser les données à jour d'un ordre, il est recommandé d'appeler OrderSelect() juste avant son utilisation.

Voir également

OrdersTotal(), OrderGetTicket(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

OrderGetInteger

Retourne la propriété demandée de l'ordre, pré-sélectionné avec OrderGetTicket ou OrderSelect. La propriété de l'ordre doit être de type datetime ou int. Il existe 2 variantes de cette fonction.

1. Retourne immédiatement la valeur de la propriété.

```
long  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référénce (le dernier paramètre).

```
bool  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id,       // Identifiant de la propriété
   long&                        long_var           // Valeur de la propriété
   );
```

Parameters

property_id

[in] Identifiant de la propriété de l'ordre. La valeur peut être l'une des valeurs de l'énumération ENUM_ORDER_PROPERTY_INTEGER

long_var

[out] Variable de type long qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type long. Si la fonction échoue, retourne 0.

Note

Ne confondez pas les ordres en attente en cours avec les positions, qui sont également affichées dans l'onglet Trading de la fenêtre Boîte à Outils.

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

Pour être sûr d'utiliser les données à jour d'un ordre, il est recommandé d'appeler OrderSelect() juste avant son utilisation.

Voir également

OrdersTotal(), OrderGetTicket(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

OrderGetString

Retourne la propriété demandée de l'ordre, pré-sélectionné avec OrderGetTicket ou OrderSelect. La propriété de l'ordre doit être de type string. Il existe 2 variantes de cette fonction.

1. Retourne immédiatement la valeur de la propriété.

```
string  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id        // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référénce (le dernier paramètre).

```
bool  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id,       // Identifiant de la propriété
   string&                     string_var         // Valeur de la propriété
   );
```

Parameters

property_id

[in] Identifiant de la propriété de l'ordre. La valeur peut être l'une des valeurs de l'énumération ENUM_ORDER_PROPERTY_STRING

string_var

[out] Variable de type string qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type string.

Note

Ne confondez pas les ordres en attente en cours avec les positions, qui sont également affichées dans l'onglet Trading de la fenêtre Boîte à Outils.

Pour le "netting" des positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), seule une position peut exister pour un même symbole à tout moment. Cette position est le résultat d'une ou plusieurs transactions. Ne confondez pas les positions avec les ordres en attente valides, qui sont également affichés dans l'onglet Trading de la fenêtre Boîte à Outils.

Si les positions individuelles sont autorisées (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes pour un même symbole.

Pour être sûr d'utiliser les données à jour d'un ordre, il est recommandé d'appeler OrderSelect() juste avant son utilisation.

Voir également

OrdersTotal(), OrderGetTicket(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

HistorySelect

Récupère l'historique des transactions et des ordres pour la période spécifiée.

```
bool  HistorySelect(
   datetime  from_date,      // Date de début
   datetime  to_date         // Date de fin
   );
```

Parameters

from_date

[in] Date de début de la requête.

to_date

[in] Date de fin de la requête.

Valeur de Retour

Retourne true en cas de succès, sinon retourne false.

Note

HistorySelect() crée une liste d'ordres et une liste de trades dans un programme mql5, pour pouvoir ensuite les utiliser dans les fonctions correspondantes. La taille de la liste des transactions peut être récupérée avec la fonction HistoryDealsTotal() ; la taille de la liste des ordres dans l'historique peut être obtenue avec HistoryOrdersTotal(). La sélection dans la liste des ordres devrait être effectuée avec HistoryOrderGetTicket(), pour les éléments de la liste des transactions, HistoryDealGetTicket() convient mieux.

Après l'utilisation de HistoryOrderSelect(), la liste des ordres disponibles de l'historique pour le programme mql5 est réinitialisée et remplie à nouveau avec l'ordre trouvé, si la recherche d'un ordre par son ticket a été effectuée avec succès. La même chose s'applique à la liste des transactions disponibles pour le programme mql5 - elle est réinitialisée avec HistoryDealSelect() et remplie à nouveau si la transaction est obtenue avec succès par son numéro de ticket.

Exemple :

```
void OnStart()
  {
   color BuyColor =clrBlue;
   color SellColor=clrRed;
//--- demande l'historique de trading
   HistorySelect(0,TimeCurrent());
//--- crée les objets
   string   name;
   uint     total=HistoryDealsTotal();
   ulong    ticket=0;
   double   price;
   double   profit;
   datetime time;
   string   symbol;
   long     type;
```

© 2000-2024, MetaQuotes Ltd.

```
   long     entry;
//--- pour toutes les transactions
   for(uint i=0;i<total;i++)
     {
      //--- essaye de récuperer le ticket des transactions
      if((ticket=HistoryDealGetTicket(i))>0)
        {
         //--- récupère les propriétés des transactions
         price =HistoryDealGetDouble(ticket,DEAL_PRICE);
         time  =(datetime)HistoryDealGetInteger(ticket,DEAL_TIME);
         symbol=HistoryDealGetString(ticket,DEAL_SYMBOL);
         type  =HistoryDealGetInteger(ticket,DEAL_TYPE);
         entry =HistoryDealGetInteger(ticket,DEAL_ENTRY);
         profit=HistoryDealGetDouble(ticket,DEAL_PROFIT);
         //--- seulement pour le symbole courant
         if(price && time && symbol==Symbol())
           {
            //--- crée l'objet prix
            name="TradeHistory_Deal_"+string(ticket);
            if(entry) ObjectCreate(0,name,OBJ_ARROW_RIGHT_PRICE,0,time,price,0,0);
            else      ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,time,price,0,0);
            //--- définit les propriétés de l'objet
            ObjectSetInteger(0,name,OBJPROP_SELECTABLE,0);
            ObjectSetInteger(0,name,OBJPROP_BACK,0);
            ObjectSetInteger(0,name,OBJPROP_COLOR,type?BuyColor:SellColor);
            if(profit!=0) ObjectSetString(0,name,OBJPROP_TEXT,"Profit: "+string(profit
           }
        }
     }
//--- applique sur le graphique
   ChartRedraw();
  }
```

Voir également

HistoryOrderSelect(), HistoryDealSelect()

© 2000-2024, MetaQuotes Ltd.

HistorySelectByPosition

Récupère l'historique des transactions et des ordres ayant l'identifiant de position spécifié.

```
bool  HistorySelectByPosition(
   long   position_id     // identifiant de la position - POSITION_IDENTIFIER
   );
```

Parameters

position_id

[in] Identifiant de la position qui est défini pour chaque ordre et transaction exécuté.

Valeur de Retour

Retourne true en cas de succès, sinon retourne false.

Note

Ne confondez pas les ordres d'un historique de trading avec les ordres en attente en cours qui apparaissent dans l'onglet "Trade" de la barre "Boîte à Outils". La liste des ordres qui ont été annulés ou qui on conduit à une transaction, est visible dans l'onglet "Historique" de la "Boîte à Outils" du terminal client.

HistorySelectByPosition() crée une liste d'ordres dans un programme mql5 et une liste de transactions ayant un identifiant de position spécifié, pour pouvoir ensuite les utiliser dans les fonctions correspondantes. La taille de la liste des transactions peut être récupérée avec la fonction HistoryDealsTotal(), la taille de la liste des ordres dans l'historique peut être obtenue avec HistoryOrdersTotal(). Pour parcourir les éléments de la liste des ordres, utilisez HistoryOrderGetTicket(), pour les éléments de la liste des transactions - HistoryDealGetTicket().

Après l'utilisation de HistoryOrderSelect(), la liste des ordres disponibles de l'historique pour le programme mql5 est réinitialisée et remplie à nouveau avec l'ordre trouvé, si la recherche d'un ordre par son ticket a été effectuée avec succès. La même chose s'applique à la liste des transactions disponibles pour le programme mql5 - elle est réinitialisée avec HistoryDealSelect() et remplie à nouveau si la transaction est obtenue avec succès par son numéro de ticket.

Voir également

HistorySelect(), HistoryOrderGetTicket(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

HistoryOrderSelect

Sélectionne un ordre dans l'historique pour l'utiliser ensuite avec les fonctions correspondantes. La fonction retourne si la fonction s'est terminé avec succès. Elle retourne false si la fonction a échoué. Pour plus de détails sur l'erreur, appelez GetLastError().

```
bool  HistoryOrderSelect(
   ulong  ticket      // Ticket de l'ordre
   );
```

Parameters

ticket

[in] Ticket de l'ordre.

Valeur de Retour

Retourne true en cas de succès, sinon false.

Note

Ne confondez pas les ordres d'un historique de trading avec les ordres en attente en cours qui apparaissent dans l'onglet "Trade" de la barre "Boîte à Outils". La liste des ordres qui ont été annulés ou qui on conduit à une transaction, est visible dans l'onglet "Historique" de la "Boîte à Outils" du terminal client.

HistoryOrderSelect() supprime dans un programme mql5 la liste des ordres de l'historique, disponibles pour être appelés, et y copie un ordre unique si HistoryOrderSelect() s'est terminée avec succès. Si vous devez parcourir toutes les transactions sélectionnées avec la fonction HistorySelect(), il est préférable d'utiliser HistoryOrderGetTicket().

Voir également

HistorySelect(), HistoryOrderGetTicket(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

HistoryOrdersTotal

Retourne le nombre d'ordres dans l'historique. Avant d'appeler HistoryOrdersTotal(), il est nécessaire de récupérer d'abord l'historique des transactions et des ordres avec les fonctions HistorySelect() ou HistorySelectByPosition().

```
int  HistoryOrdersTotal();
```

Valeur de Retour

Valeur de type int.

Note

Ne confondez pas les ordres d'un historique de trading avec les ordres en attente en cours qui apparaissent dans l'onglet "Trade" de la barre "Boîte à Outils". La liste des ordres qui ont été annulés ou qui on conduit à une transaction, est visible dans l'onglet "Historique" de la "Boîte à Outils" du terminal client.

Voir également

HistorySelect(), HistoryOrderSelect(), HistoryOrderGetTicket(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

HistoryOrderGetTicket

Retoure le ticket de l'ordre correspondant dans l'historique. Avant d'appeler HistoryOrderGetTicket(), il est nécessaire de récupérer d'abord l'historique des transactions et des ordres avec les fonctions HistorySelect() ou HistorySelectByPosition().

```
ulong  HistoryOrderGetTicket(
   int  index      // indice dans la liste des ordres
   );
```

Parameters

index

[in] Numéro de l'ordre dans la liste des ordres

Valeur de Retour

Valeur de type ulong. Si la fonction échoue, retourne 0.

Note

Ne confondez pas les ordres d'un historique de trading avec les ordres en attente en cours qui apparaissent dans l'onglet "Trade" de la barre "Boîte à Outils". La liste des ordres qui ont été annulés ou qui on conduit à une transaction, est visible dans l'onglet "Historique" de la "Boîte à Outils" du terminal client.

Exemple :

```
void OnStart()
  {
   datetime from=0;
   datetime to=TimeCurrent();
//--- demande l'historique en entier
   HistorySelect(from,to);
//--- variables pour retourner les valeurs des propriétés de l'ordre
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   datetime time_done;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- nombre d'ordres en attente en cours
   uint     total=HistoryOrdersTotal();
//--- parcours les ordres dans une boucle
   for(uint i=0;i<total;i++)
     {
      //--- retourne le ticket de l'ordre par sa position dans la liste
      if((ticket=HistoryOrderGetTicket(i))>0)
        {
```

© 2000-2024, MetaQuotes Ltd.

```
        //--- retourne les propriétés de l'ordre
        open_price    =HistoryOrderGetDouble(ticket,ORDER_PRICE_OPEN);
        time_setup    =(datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_SETUP);
        time_done     =(datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_DONE);
        symbol        =HistoryOrderGetString(ticket,ORDER_SYMBOL);
        order_magic   =HistoryOrderGetInteger(ticket,ORDER_MAGIC);
        positionID    =HistoryOrderGetInteger(ticket,ORDER_POSITION_ID);
        initial_volume=HistoryOrderGetDouble(ticket,ORDER_VOLUME_INITIAL);
        type          =GetOrderType(HistoryOrderGetInteger(ticket,ORDER_TYPE));
        //--- prépare et affiche les informations de l'ordre
        printf("#ticket %d %s %G %s à %G a été établi à %s => fait à %s, pos ID=%d",
               ticket,                  // ticket de l'ordre
               type,                    // type
               initial_volume,          // volume placé
               symbol,                  // symbole
               open_price,              // prix d'ouverture spécifié
               TimeToString(time_setup),// heure du placement de l'ordre
               TimeToString(time_done), // heure d'exécution de l'ordre ou de sa supp
               positionID               // identifiant de la position à laquelle la t
               );
      }
    }
//---
  }
//+------------------------------------------------------------------+
//| Retourne le nom du type de l'ordre                               |
//+------------------------------------------------------------------+
string GetOrderType(long type)
  {
   string str_type="opération inconnue";
   switch(type)
     {
      case (ORDER_TYPE_BUY):            return("buy");
      case (ORDER_TYPE_SELL):           return("sell");
      case (ORDER_TYPE_BUY_LIMIT):      return("buy limit");
      case (ORDER_TYPE_SELL_LIMIT):     return("sell limit");
      case (ORDER_TYPE_BUY_STOP):       return("buy stop");
      case (ORDER_TYPE_SELL_STOP):      return("sell stop");
      case (ORDER_TYPE_BUY_STOP_LIMIT): return("buy stop limit");
      case (ORDER_TYPE_SELL_STOP_LIMIT):return("sell stop limit");
     }
   return(str_type);
  }
```

Voir également

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

HistoryOrderGetDouble

Retourne la propriété demandée de l'ordre. La propriété de l'ordre doit être de type double. Il existe 2 variantes de la fonction.

1. Retourne immédiatement la valeur de la propriété.

```
double  HistoryOrderGetDouble(
   ulong                        ticket_number,      // Ticket
   ENUM_ORDER_PROPERTY_DOUBLE   property_id         // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référénce (le dernier paramètre).

```
bool  HistoryOrderGetDouble(
   ulong                        ticket_number,      // Ticket
   ENUM_ORDER_PROPERTY_DOUBLE   property_id,        // Identifiant de la propriété
   double&                      double_var          // Valeur de la propriété
   );
```

Parameters

ticket_number

[in] Ticket de l'ordre.

property_id

[in] Identifiant de la propriété de l'ordre. La valeur peut être l'une des valeurs de l'énumération ENUM_ORDER_PROPERTY_DOUBLE

double_var

[out] Variable de type double qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type double.

Note

Ne confondez pas les ordres d'un historique de trading avec les ordres en attente en cours qui apparaissent dans l'onglet "Trade" de la barre "Boîte à Outils". La liste des ordres qui ont été annulés ou qui on conduit à une transaction, est visible dans l'onglet "Historique" de la "Boîte à Outils" du terminal client.

Voir également

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

HistoryOrderGetInteger

Retourne la propriété demandée de l'ordre. La propriété de l'ordre doit être de type datetime ou int. Il existe 2 variantes de la fonction.

1. Retourne immédiatement la valeur de la propriété.

```
long  HistoryOrderGetInteger(
   ulong                        ticket_number,      // Ticket
   ENUM_ORDER_PROPERTY_INTEGER  property_id         // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référence (le dernier paramètre).

```
bool  HistoryOrderGetInteger(
   ulong                        ticket_number,      // Ticket
   ENUM_ORDER_PROPERTY_INTEGER  property_id,        // Identifiant de la propriété
   long&                        long_var            // Valeur de la propriété
   );
```

Parameters

ticket_number

[in] Ticket de l'ordre.

property_id

[in] Identifiant de la propriété de l'ordre. La valeur peut être l'une des valeurs de l'énumération ENUM_ORDER_PROPERTY_INTEGER

long_var

[out] Variable de type long qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type long.

Note

Ne confondez pas les ordres d'un historique de trading avec les ordres en attente en cours qui apparaissent dans l'onglet "Trade" de la barre "Boîte à Outils". La liste des ordres qui ont été annulés ou qui on conduit à une transaction, est visible dans l'onglet "Historique" de la "Boîte à Outils" du terminal client.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction Trade                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- récupère le ticket du dernier ordre depuis l'historique de trading de la semaine
   ulong last_order=GetLastOrderTicket();
```

© 2000-2024, MetaQuotes Ltd.

```
   if(HistoryOrderSelect(last_order))
     {
      //--- heure de placement d'un ordre en millisecondes depuis le 01.01.1970
      long time_setup_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_SETUP_MSC);
      PrintFormat("Ordre #%d ORDER_TIME_SETUP_MSC=%i64 => %s",
                  last_order,time_setup_msc,TimeToString(time_setup_msc/1000));
      //--- heure d'éxécution/annulation d'un ordre en millisecondes depuis le 01.01.1
      long  time_done_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_DONE_MSC);
      PrintFormat("Ordre #%d ORDER_TIME_DONE_MSC=%i64 => %s",
                  last_order,time_done_msc,TimeToString(time_done_msc/1000));
     }
   else // notification de l'échec
      PrintFormat("Echec de HistoryOrderSelect() pour #%d. Code d'erreur=%d",
                  last_order,GetLastError());

//---
  }
//+------------------------------------------------------------------+
//| Retourne le ticket du dernier ordre dans l'historique ou -1      |
//+------------------------------------------------------------------+
ulong GetLastOrderTicket()
  {
//--- récupère l'historique des 7 derniers jours
   if(!GetTradeHistory(7))
     {
      //--- notification d'un appel non réussi et retourne -1
      Print(__FUNCTION__," HistorySelect() a retourné false");
      return -1;
     }
//---
   ulong first_order,last_order,orders=HistoryOrdersTotal();
//--- utilise les ordres s'il y en a
   if(orders>0)
     {
      Print("Ordres = ",orders);
      first_order=HistoryOrderGetTicket(0);
      PrintFormat("first_order = %d",first_order);
      if(orders>1)
        {
         last_order=HistoryOrderGetTicket((int)orders-1);
         PrintFormat("last_order = %d",last_order);
         return last_order;
        }
      return first_order;
     }
//--- aucun ordre trouvé, retourne -1
   return -1;
  }
//+------------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Récupère l'historique des derniers jours et retourne false en cas d'échec|
//+------------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- utilise la période de la semaine pour récupérer l'historique de trading
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- effectue une demande et vérifie le résultat
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Code d'erreur=",GetLastError());
      return false;
     }
//--- l'historique a été récupéré avec succès
   retourne vrai;
  }
```

Voir également

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

HistoryOrderGetString

Retourne la propriété demandée de l'ordre. La propriété de l'ordre doit être de type string. Il existe 2 variantes de la fonction.

1. Retourne immédiatement la valeur de la propriété.

```
string  HistoryOrderGetString(
   ulong                        ticket_number,      // Ticket
   ENUM_ORDER_PROPERTY_STRING   property_id         // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référénce (le dernier paramètre).

```
bool  HistoryOrderGetString(
   ulong                        ticket_number,      // Ticket
   ENUM_ORDER_PROPERTY_STRING   property_id,        // Identifiant de la propriété
   string&                      string_var          // Valeur de la propriété
   );
```

Parameters

ticket_number

[in] Ticket de l'ordre.

property_id

[in] Identifiant de la propriété de l'ordre. La valeur peut être l'une des valeurs de l'énumération ENUM_ORDER_PROPERTY_STRING

string_var

[out] Variable de type string.

Valeur de Retour

Valeur de type string.

Note

Ne confondez pas les ordres d'un historique de trading avec les ordres en attente en cours qui apparaissent dans l'onglet "Trade" de la barre "Boîte à Outils". La liste des ordres qui ont été annulés ou qui on conduit à une transaction, est visible dans l'onglet "Historique" de la "Boîte à Outils" du terminal client.

Voir également

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriétés d'un Ordre

© 2000-2024, MetaQuotes Ltd.

HistoryDealSelect

Sélectionne une transaction dans l'historique pour l'utiliser ensuite avec les fonctions correspondantes. La fonction retourne si la fonction s'est terminé avec succès. Elle retourne false si la fonction a échoué. Pour plus de détails sur l'erreur, appelez GetLastError().

```
bool  HistoryDealSelect(
   ulong  ticket      // ticket de la transaction
   );
```

Parameters

ticket

[in] Ticket de la transaction.

Valeur de Retour

Retourne true en cas de succès, sinon false.

Note

Ne confondez pas les ordres, les transactions et les positions. Chaque transaction est le résultat de l'exécution d'un ordre, chaque position est le résultat global d'une ou plusieurs transactions.

HistoryDealSelect() supprime dans un programme mql5 la liste des transactions disponibles pour référence et copie la transaction unique si HistoryDealSelect() s'est terminée avec succès. Si vous devez parcourir toutes les transactions sélectionnées avec la fonction HistorySelect(), il est préférable d'utiliser HistoryDealGetTicket().

Voir également

HistorySelect(), HistoryDealGetTicket(), Propriétés d'une transaction

© 2000-2024, MetaQuotes Ltd.

HistoryDealsTotal

Retourne le nombre de transactions dans l'historique. Avant d'appeler HistoryDealsTotal(), il est nécessaire de récupérer d'abord l'historique des transactions et des ordres avec les fonctions HistorySelect() ou HistorySelectByPosition().

```
int  HistoryDealsTotal();
```

Valeur de Retour

Valeur de type int.

Note

Ne confondez pas les ordres, les transactions et les positions. Chaque transaction est le résultat de l'exécution d'un ordre, chaque position est le résultat global d'une ou plusieurs transactions.

Voir également

HistorySelect(), HistoryDealGetTicket(), Propriétés d'une transaction

© 2000-2024, MetaQuotes Ltd.

HistoryDealGetTicket

La fonction sélectionne une transaction pour pouvoir l'utiliser ensuite et retourne le ticket de la transaction de l'historique. Avant d'appeler HistoryDealGetTicket(), il est nécessaire de récupérer d'abord l'historique des transactions et des ordres avec les fonctions HistorySelect() ou HistorySelectByPosition().

```
ulong  HistoryDealGetTicket(
   int  index      // indice de la transaction
   );
```

Parameters

index

[in] Numéro d'une transaction dans la liste des transactions

Valeur de Retour

Valeur de type ulong. Si la fonction échoue, retourne 0.

Note

Ne confondez pas les ordres, les transactions et les positions. Chaque transaction est le résultat de l'exécution d'un ordre, chaque position est le résultat global d'une ou plusieurs transactions.

Exemple :

```
void OnStart()
  {
   ulong deal_ticket;            // ticket de la transaction
   ulong order_ticket;           // ticket de l'ordre sur lequel la transaction a été
   datetime transaction_time;    // heure d'exécution de la transaction
   long deal_type ;              // type de l'opération de trading
   long position_ID;             // identifiant de la position
   string deal_description;      // description de l'opération
   double volume;                // volume de l'opération
   string symbol;                // symbole de la transaction
//--- définit la date de début et de fin pour demander l'historique des transactions
   datetime from_date=0;         // date de début
   datetime to_date=TimeCurrent();// date de fin
//--- demande l'historique des transactions pour la période spécifiée
   HistorySelect(from_date,to_date);
//--- nombre total dans la liste des transactions
   int deals=HistoryDealsTotal();
//--- traite maintenant chaque trade
   for(int i=0;i<deals;i++)
     {
      deal_ticket=               HistoryDealGetTicket(i);
      volume=                    HistoryDealGetDouble(deal_ticket,DEAL_VOLUME);
      transaction_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
      order_ticket=              HistoryDealGetInteger(deal_ticket,DEAL_ORDER);
      deal_type=                 HistoryDealGetInteger(deal_ticket,DEAL_TYPE);
```

© 2000-2024, MetaQuotes Ltd.

```
      symbol=                    HistoryDealGetString(deal_ticket,DEAL_SYMBOL);
      position_ID=               HistoryDealGetInteger(deal_ticket,DEAL_POSITION_ID);
      deal_description=          GetDealDescription(deal_type,volume,symbol,order_tic
      //--- affiche correctement le numéro de la transaction
      string print_index=StringFormat("% 3d",i);
      //--- affiche les informations sur la transaction
      Print(print_index+" : deal #",deal_ticket," à ",transaction_time,deal_descriptio
     }
  }
//+------------------------------------------------------------------+
//| Retourne la description textuelle de l'opération                 |
//+------------------------------------------------------------------+
string GetDealDescription(long deal_type,double volume,string symbol,long ticket,long
  {
   string descr;
//---
   switch(deal_type)
     {
      case DEAL_TYPE_BALANCE:                  return ("solde");
      case DEAL_TYPE_CREDIT:                   return ("crédit");
      case DEAL_TYPE_CHARGE:                   return ("facturé");
      case DEAL_TYPE_CORRECTION:               return ("correction");
      case DEAL_TYPE_BUY:                      descr="buy"; break;
      case DEAL_TYPE_SELL:                     descr="sell"; break;
      case DEAL_TYPE_BONUS:                    return ("bonus");
      case DEAL_TYPE_COMMISSION:               return ("commission supplémentaire");
      case DEAL_TYPE_COMMISSION_DAILY:         return ("commission quotidienne");
      case DEAL_TYPE_COMMISSION_MONTHLY:       return ("commission mnensuelle");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY:   return ("commission quotidienne de l'ag
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return ("commission mensuelle de l'agen
      case DEAL_TYPE_INTEREST:                 return ("taux d'intérêt");
      case DEAL_TYPE_BUY_CANCELED:             descr="transaction buy annulée"; break;
      case DEAL_TYPE_SELL_CANCELED:            descr="transaction sell annulée"; break
     }
   descr=StringFormat("%s %G %s (ordre #%d, identifiant de la position %d)",
                      descr,  // description en cours
                      volume, // volume de la transaction
                      symbol, // symbol de la transaction
                      ticket, // ticket de l'ordre associé à la transaction
                      pos_ID  // identifiant de la position à laquelle la transaction
                      );
   return(descr);
//---
  }
```

Voir également

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), Propriétés d'une transaction

© 2000-2024, MetaQuotes Ltd.

HistoryDealGetDouble

Retourne la propriété demandée d'une transaction. La propriété de la transaction doit être de type double. Il existe 2 variantes de la fonction.

1. Retourne immédiatement la valeur de la propriété.

```
double  HistoryDealGetDouble(
   ulong                      ticket_number,      // Ticket
   ENUM_DEAL_PROPERTY_DOUBLE  property_id         // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référence (le dernier paramètre).

```
bool  HistoryDealGetDouble(
   ulong                      ticket_number,      // Ticket
   ENUM_DEAL_PROPERTY_DOUBLE  property_id,        // Identifiant de la propriété
   double&                    double_var          // Valeur de la propriété
   );
```

Parameters

ticket_number

[in] Ticket de la transaction.

property_id

[in] Identifiant de la propriété de la transaction. La valeur peut être l'une des valeurs de l'énumération ENUM_DEAL_PROPERTY_DOUBLE

double_var

[out] Variable de type double qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type double.

Note

Ne confondez pas les ordres, les transactions et les positions. Chaque transaction est le résultat de l'exécution d'un ordre, chaque position est le résultat global d'une ou plusieurs transactions.

Voir également

HistorySelect(), HistoryDealsTotal(), HistoryDealGetTicket(), Propriétés d'une transaction

© 2000-2024, MetaQuotes Ltd.

HistoryDealGetInteger

Retourne la propriété demandée d'une transaction. La propriété de la transaction doit être de type datetime ou int. Il existe 2 variantes de la fonction.

1. Retourne immédiatement la valeur de la propriété.

```
long  HistoryDealGetInteger(
   ulong                        ticket_number,      // Ticket
   ENUM_DEAL_PROPERTY_INTEGER   property_id         // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référence (le dernier paramètre).

```
bool  HistoryDealGetInteger(
   ulong                        ticket_number,      // Ticket
   ENUM_DEAL_PROPERTY_INTEGER   property_id,        // Identifiant de la propriété
   long&                        long_var            // Valeur de la propriété
   );
```

Parameters

ticket_number

[in] Ticket du trade.

property_id

[in] Identifiant de la propriété de la transaction. La valeur peut être l'une des valeurs de l'énumération ENUM_DEAL_PROPERTY_INTEGER

long_var

[out] Variable de type long qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type long.

Note

Ne confondez pas les ordres, les transactions et les positions. Chaque transaction est le résultat de l'exécution d'un ordre, chaque position est le résultat global d'une ou plusieurs transactions.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction Trade                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- récupère le ticket de la dernière transaction depuis l'historique de trading de
   ulong last_deal=GetLastDealTicket();
   if(HistoryDealSelect(last_deal))
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- heure d'exécution de la transaction en millisecondes depuis le 01.01.1970
      long deal_time_msc=HistoryDealGetInteger(last_deal,DEAL_TIME_MSC);
      PrintFormat("Transaction #%d DEAL_TIME_MSC=%i64 => %s",
                  last_deal,deal_time_msc,TimeToString(deal_time_msc/1000));
     }
   else
      PrintFormat("Echec de HistoryDealSelect() pour #%d. Code d'erreur=%d",
                  last_deal,GetLastError());
//---
  }
//+------------------------------------------------------------------+
//| Retourne le ticket de la dernière transaction ou -1              |
//+------------------------------------------------------------------+
ulong GetLastDealTicket()
  {
//--- récupère l'historique des 7 derniers jours
   if(!GetTradeHistory(7))
     {
      //--- notification d'un appel non réussi et retourne -1
      Print(__FUNCTION__," HistorySelect() a retourné false");
      return -1;
     }
//---
   ulong first_deal,last_deal,deals=HistoryOrdersTotal();
//--- utilise les ordres s'il y en a
   if(deals>0)
     {
      Print("Transactions = ",deals);
      first_deal=HistoryDealGetTicket(0);
      PrintFormat("first_deal = %d",first_deal);
      if(deals>1)
        {
         last_deal=HistoryDealGetTicket((int)deals-1);
         PrintFormat("last_deal = %d",last_deal);
         return last_deal;
        }
      return first_deal;
     }
//--- aucune transaction trouvée, retourne -1
   return -1;
  }
//+------------------------------------------------------------------------+
//| Récupère l'historique des derniers jours et retourne false en cas d'échec|
//+------------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- utilise la période de la semaine pour récupérer l'historique de trading
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
```

```
   ResetLastError();
//--- effectue une demande et vérifie le résultat
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Code d'erreur=",GetLastError());
      return false;
     }
//--- l'historique a été récupéré avec succès
   retourne vrai;
  }
```

Voir également

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Propriétés d'une transaction

© 2000-2024, MetaQuotes Ltd.

HistoryDealGetString

Retourne la propriété demandée d'une transaction. La propriété de la transaction doit être de type string. Il existe 2 variantes de la fonction.

1. Retourne immédiatement la valeur de la propriété.

```
string  HistoryDealGetString(
   ulong                       ticket_number,      // Ticket
   ENUM_DEAL_PROPERTY_STRING   property_id         // Identifiant de la propriété
   );
```

2. Retourne true ou false, suivant le succès de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable correspondante passée par référénce (le dernier paramètre).

```
bool  HistoryDealGetString(
   ulong                       ticket_number,      // Ticket
   ENUM_DEAL_PROPERTY_STRING   property_id,        // Identifiant de la propriété
   string&                     string_var          // Valeur de la propriété
   );
```

Parameters

ticket_number

[in] Ticket de la transaction.

property_id

[in] Identifiant de la propriété de la transaction. La valeur peut être l'une des valeurs de l'énumération ENUM_DEAL_PROPERTY_STRING

string_var

[out] Variable de type string qui prendra la valeur de la propriété demandée.

Valeur de Retour

Valeur de type string.

Note

Ne confondez pas les ordres, les transactions et les positions. Chaque transaction est le résultat de l'exécution d'un ordre, chaque position est le résultat global d'une ou plusieurs transactions.

Voir également

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Propriétés d'une transaction

© 2000-2024, MetaQuotes Ltd.

Signaux de Trading

Ceci est le groupe de fonctions destinés à la gestion des signaux de trading. Les fonctions permettent de :

- obtenir des informations sur les signaux de trading disponibles pour la copie,
- obtenir et définir les paramètres de copie du signal,
- s'abonner et se désabonner de la copie du signal en utilisant les fonctions de MQL5.

Fonction	Action
SignalBaseGetDouble	Retourne la valeur de la propriété de type double du signal sélectionné
SignalBaseGetInteger	Retourne la valeur de la propriété de type integer du signal sélectionné
SignalBaseGetString	Retourne la valeur de la propriété de type chaîne de caractères du signal sélectionné.
SignalBaseSelect	Sélectionne un signal à partir des signaux disponibles dans le terminal pour travailler par la suite avec ce signal
SignalBaseTotal	Retourne le nombre total de signaux disponibles dans le terminal
SignalInfoGetDouble	Retourne la valeur de la propriété de type double des paramètres de copie du signal
SignalInfoGetInteger	Retourne la valeur de la propriété de type integer des paramètres de copie du signal
SignalInfoGetString	Retourne la valeur de la propriété de type chaîne de carctères des paramètres de copie du signal.
SignalInfoSetDouble	Définit la valeur de la propriété de type double des paramètres de copie du signal
SignalInfoSetInteger	Définit la valeur de la propriété de type integer des paramètres de copie du signal
SignalSubscribe	Souscrit au signal de trading
SignalUnsubscribe	Annule l'abonnement

© 2000-2024, MetaQuotes Ltd.

SignalBaseGetDouble

Retourne la valeur de la propriété de type double pour le signal sélectionné.

```
double  SignalBaseGetDouble(
   ENUM_SIGNAL_BASE_DOUBLE     property_id,     // identifiant de la propriété
   );
```

Paramètres

property_id

[in] Identifiant de la propriété du signal. La valeur peut être l'une des valeurs de l'énumération ENUM_SIGNAL_BASE_DOUBLE.

Valeur de Retour

La valeur de type double de la propriété du signal sélectionné.

© 2000-2024, MetaQuotes Ltd.

SignalBaseGetInteger

Retourne la valeur de la propriété de type integer du signal sélectionné.

```
long  SignalBaseGetInteger(
   ENUM_SIGNAL_BASE_INTEGER     property_id,     // identifiant de la propriété
   );
```

Paramètres

property_id

[in] Identifiant de la propriété du signal. La valeur peut être l'une des valeurs de l'énumération ENUM_SIGNAL_BASE_INTEGER.

Valeur de Retour

La valeur de type integer de la propriété du signal sélectionné.

© 2000-2024, MetaQuotes Ltd.

SignalBaseGetString

Retourne la valeur de la propriété de type chaîne de caractères du signal sélectionné.

```
string  SignalBaseGetString(
   ENUM_SIGNAL_BASE_STRING     property_id,     // identifiant de la propriété
   );
```

Paramètres

property_id

[in] Identifiant de la propriété du signal. La valeur peut être l'une des valeurs de l'énumération ENUM_SIGNAL_BASE_STRING.

Valeur de Retour

La valeur de la propriété de type de chaîne de caractères du signal sélectionné.

© 2000-2024, MetaQuotes Ltd.

SignalBaseSelect

Sélectionne un signal à partir des signaux disponibles dans le terminal pour travailler avec.

```
bool  SignalBaseSelect(
   int     index     // indice du signal
   );
```

Paramètres

index

[in] Indice du signal dans la base des signaux de trading.

Valeur de Retour

Retourne vrai en cas de succès, sinon retourne faux. Pour en savoir plus au sujet de l'erreur, appeller la fonction GetLastError().

Exemple :

```
void OnStart()
  {
//--- récupère le nombre total de signaux dans le terminal
   int total=SignalBaseTotal();
//--- traitement de tous les signaux
   for(int i=0;i<total;i++)
     {
      //--- sélection du signal par l'index
      if(SignalBaseSelect(i))
        {
         //--- obtention des propriétés du signal
         long   id    =SignalBaseGetInteger(SIGNAL_BASE_ID);          // id du signal
         long   pips  =SignalBaseGetInteger(SIGNAL_BASE_PIPS);        // profit en pip
         long   subscr=SignalBaseGetInteger(SIGNAL_BASE_SUBSCRIBERS); // nombre d'abon
         string name  =SignalBaseGetString(SIGNAL_BASE_NAME);         // nom du signal
         double price =SignalBaseGetDouble(SIGNAL_BASE_PRICE);        // prix du signa
         string curr  =SignalBaseGetString(SIGNAL_BASE_CURRENCY);     // devise du sig
         //--- affiche tous les signaux rentables gratuits avec les abonnés
         if(price==0.0 && pips>0 && subscr>0)
            PrintFormat("id=%d, name=\"%s\", currency=%s, pips=%d, subscribers=%d",id,
        }
      else PrintFormat("Erreur dans l'appel du SignalBaseSelect. Error code=%d",GetLas
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

SignalBaseTotal

Retourne le nombre total de signaux disponibles dans le terminal.

```
int  SignalBaseTotal();
```

Valeur de Retour

Le nombre total de signaux disponibles dans le terminal.

© 2000-2024, MetaQuotes Ltd.

SignalInfoGetDouble

Retourne la valeur de la propriété de type double des paramètres de copie du signal.

```
double  SignalInfoGetDouble(
   ENUM_SIGNAL_INFO_DOUBLE     property_id,     // identifiant de la propriété
   );
```

Paramètres

property_id

[in] Identifiant des paramètres de copie du signal. La valeur peut être l'une des valeurs de l'énumération ENUM_SIGNAL_INFO_DOUBLE.

Valeur de Retour

La valeur de la propriété de type double des paramètres de copie du signal.

© 2000-2024, MetaQuotes Ltd.

SignalInfoGetInteger

Retourne la valeur de la propriété de type inetger des paramètres de copie du signal.

```
long  SignalInfoGetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // identifiant de la propriété
   );
```

Paramètres

property_id

[in] Identifiant des paramètres de copie du signal. La valeur peut être l'une des valeurs de l'énumération ENUM_SIGNAL_INFO_INTEGER.

Valeur de Retour

La valeur de la propriété de type integer des paramètres de copie du signal.

© 2000-2024, MetaQuotes Ltd.

SignalInfoGetString

Retourne la valeur de la propriété de type chaîne de caractères des paramètres de copie du signal.

```
string  SignalInfoGetString(
   ENUM_SIGNAL_INFO_STRING     property_id,     // identifiant de la propriété
   );
```

Paramètres

property_id

[in] Identifiant des paramètres de copie du signal. La valeur peut être l'une des valeurs de l'énumération ENUM_SIGNAL_INFO_STRING.

Valeur de Retour

La valeur de la propriété de type dechaîne de caractères des paramètres de copie du signal.

© 2000-2024, MetaQuotes Ltd.

SignalInfoSetDouble

Définit la valeur de la propriété de type double des paramètres de copie du signal.

```
bool  SignalInfoSetDouble(
   ENUM_SIGNAL_INFO_DOUBLE      property_id,     // identifiant de la propriété
   double                       value            // nouvelle valeur
   );
```

Paramètres

property_id

[in] Identifiant des paramètres de copie du signal. La valeur peut être l'une des valeurs de l'énumération ENUM_SIGNAL_INFO_DOUBLE.

value

[in] La valeur de la propriété des paramètres de copie du signal.

Valeur de Retour

Retourne vrai si la propriété a été modifiée, sinon retourne faux. Pour en savoir plus au sujet de l'erreur, appeller la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

SignalInfoSetInteger

Définit la valeur de la propriété de type integer des paramètres de copie du signal.

```
bool  SignalInfoSetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // identifiant de la propriété
   long                         value            // nouvelle valeur
   );
```

Paramètres

property_id

[in] Identifiant des paramètres de copie du signal. La valeur peut être l'une des valeurs de l'énumération ENUM_SIGNAL_INFO_INTEGER.

value

[in] La valeur de la propriété des paramètres de copie du signal.

Valeur de Retour

Retourne vrai si la propriété a été modifiée, sinon retourne faux. Pour en savoir plus au sujet de l'erreur, appeller la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

SignalSubscribe

Souscrit au signal de trading.

```
bool  SignalSubscribe(
   long     signal_id     // identifiant du signal
   );
```

Paramètres

signal_id

[in] Identifiant du signal.

Valeur de Retour

Retourne vrai si l'abonnement réussi, sinon retourne faux. Pour en savoir plus au sujet de l'erreur, appeller la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

SignalUnsubscribe

Annule l'abonnement.

```
bool  SignalUnsubscribe();
```

Valeur de Retour

Retourne vrai si l'abonnement a été annulé avec succès, sinon retourne faux. Pour en savoir plus au sujet de l'erreur, appeller la fonction GetLastError().

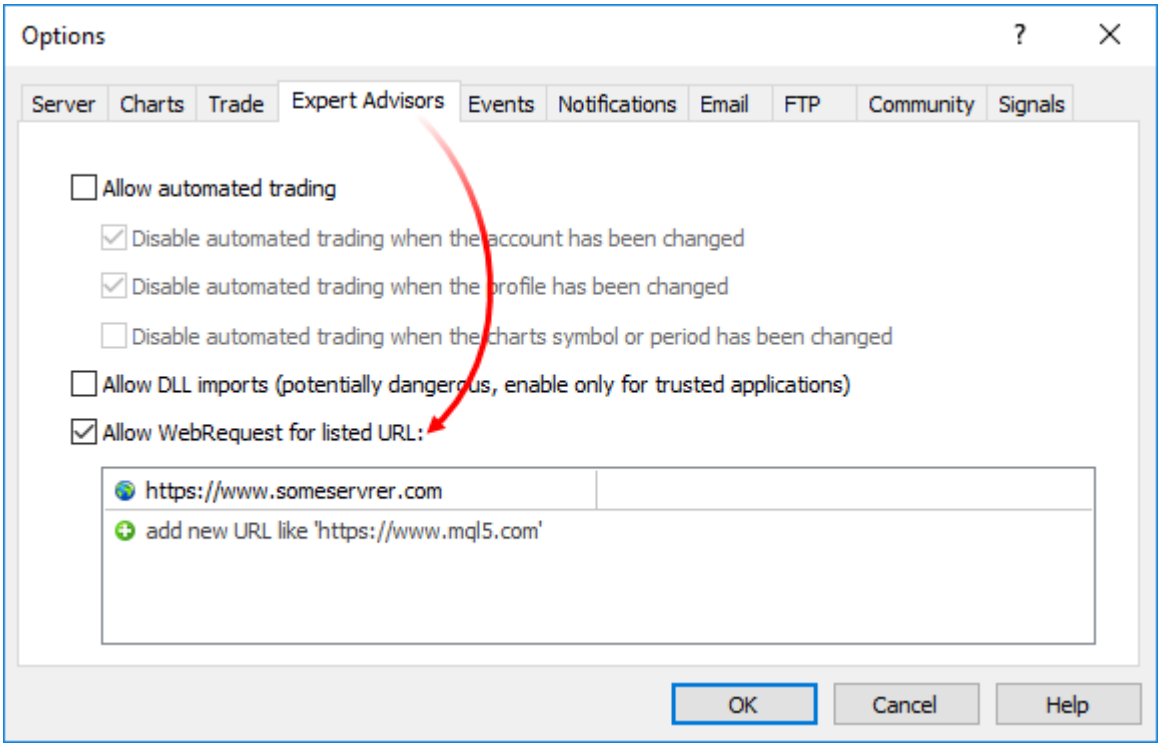
© 2000-2024, MetaQuotes Ltd.

Fonctions réseau

Les programmes MQL5 peuvent échanger des données avec des serveurs distants, ainsi qu'envoyer des notifications push, des emails et des données via FTP.

- Le groupe de fonctions Socket* permet d'établir une connexion TCP (incluant une connexion TLS sécurisée) avec un hôte remote via les sockets système. Le principe d'opération est simple : création d'une socket, connexion au serveur et début de la lecture et de l'écriture des données.
- La fonction WebRequest est conçue pour travailler avec les ressources web et permet d'envoyer facilement des requêtes HTTP (incluant GET et POST).
- SendFTP, SendMail et SendNotification sont des fonctions plus simples pour envoyer des fichiers, des emails et des notifications mobiles.

Pour la sécurité de l'utilisateur finale, la liste des adresses IP autorisées est implémentée du côté du terminal client. La liste contient les adresses IP auxquelles le programme MQL5 est autorisé de se connecter via les fonctions Socket* et WebRequest. Par exemple, si le programme doit se connecter à https://www.someserver.com, cette adresse doit être indiquée explicitement par un utilisateur du terminal dans la liste. Une adresse ne peut pas être ajoutée programmatiquement.



Ajout d'un message explicite au programme MQL5 pour notifier à l'utilisateur le besoin d'une configuration additionnelle. Vous pouvez le faire via #property description, Alert ou Print.

Fonction	Action
SocketCreate	Crée une socket avec les flags spécifiés et retourne son handle
SocketClose	Ferme une socket
SocketConnect	Connexion au serveur avec un contrôle du timeout
SocketIsConnected	Vérifie si la socket est actuellement connectée

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
SocketIsReadable	Retourne le nombre d'octets pouvant être lus sur une socket
SocketIsWritable	Vérifie si des données peuvent être écrites actuellement sur une socket
SocketTimeouts	Définit les timeouts pour recevoir et envoyer des données sur une socket système
SocketRead	Lit les données d'une socket
SocketSend	Ecrit des données sur une socket
SocketTlsHandshake	Initie une connexion TLS (SSL) vers un hôte spécifié via le protocole TLS Handshake
SocketTlsCertificate	Retourne les données sur le certificat utilisé pour une connexion sécurisée au réseau
SocketTlsRead	Lit des données depuis une connexion TLS sécurisée
SocketTlsReadAvailable	Lit toutes les données disponibles depuis une connexion TLS sécurisée
SocketTlsSend	Envoi des données via une connexion TLS sécurisée
WebRequest	Envoi une requête HTTP vers un serveur spécifié
SendFTP	Envoi un fichier à une adresse spécifiée dans l'onglet FTP
SendMail	Envoi un email à une adresse spécifiée dans l'onglet Email de la fenêtre Options
SendNotification	Envoi des notifications push vers des terminaux mobiles dont les MetaQuotes IDs sont spécifiés dans l'onglet Notifications

© 2000-2024, MetaQuotes Ltd.

SocketCreate

Crée une socket avec les flags spécifiés et retourne son handle.

```
int  SocketCreate(
   uint     flags          // flags
   );
```

Parameters

flags

[in] Combinaison de flags définissant le mode d'utilisation d'une socket. Actuellement, seul un flag est supporté — SOCKET_DEFAULT.

Valeur de Retour

En cas de création réussie d'une socket, retourne son handle, sinon INVALID_HANDLE.

Notes

Pour libérér la mémoire de l'ordinateur d'une socket non utilisée, appelez SocketClose.

Vous pouvez créer un maximum de 128 sockets depuis un seul programme MQL5. Si la limite est dépassée, l'erreur 5271 (ERR_NETSOCKET_TOO_MANY_OPENED) est écrite dans _LastError.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 - "Function is not allowed for call".

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Ajoute une adresse à la liste des adresses autorisées dans les
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Envoi une commande au serveur                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si la connexion TLS sécurisée est utilisée via le port 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si la connexion TCP standard est utilisée
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lit la réponse du serveur                                        |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lit les données des sockets tant qu'il y en a et juqu'au timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- différentes commandes de lecture suivant que la connexion est sécurisée
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analyse la réponse
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- n'affiche que l'en-tete de la réponse
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("En-tete de la réponse HTTP reçue :");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
```

```
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- vérifie le handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connexion si tout est ok
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connexion établie avec ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la connexion est sécurisée par un certificat, affiche ses données
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificat TLS :");
            Print("   Propriétaire :  ",subject);
            Print("   Emetteur :  ",issuer);
            Print("   Numéro :     ",serial);
            Print("   Print : ",thumbprint);
            Print("   Expiration : ",expiration);
            ExtTLS=true;
           }
         //--- envoi une requete GET au serveur
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Requête GET envoyée");
            //--- lit la réponse
            if(!HTTPRecv(socket,1000))
               Print("Echec d'obtention d'une réponse, erreur ",GetLastError());
           }
         else
            Print("Echec d'envoi de la requete GET, erreur ",GetLastError());
        }
      else
        {
         Print("Connexion à ",Address,":",Port," échouée, erreur ",GetLastError());
        }
      //--- ferme la socket après son utilisation
      SocketClose(socket);
     }
   else
      Print("Echec de création de la socket, erreur ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

SocketClose

Ferme une socket.

```
bool  SocketClose(
   const int  socket      // handle de la socket
   );
```

Parameters

socket

[in] Handle d'une socket à fermer. Le handle est retourné par la fonction SocketCreate. Lorsqu'un handle incorrect est passé, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est écrite dans _LastError.

Valeur de Retour

Retourne true en cas de succès, sinon false.

Note

Si une connexion via SocketConnect avait été créée auparavant pour une socket, elle est interrompue.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 - "Function is not allowed for call".

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Ajoute une adresse à la liste des adresses autorisées dans les
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Envoi une commande au serveur                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si la connexion TLS sécurisée est utilisée via le port 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si la connexion TCP standard est utilisée
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lit la réponse du serveur                                        |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lit les données des sockets tant qu'il y en a et juqu'au timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- différentes commandes de lecture suivant que la connexion est sécurisée
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analyse la réponse
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- n'affiche que l'en-tete de la réponse
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("En-tete de la réponse HTTP reçue :");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
```

```
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- vérifie le handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connexion si tout est ok
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connexion établie avec ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la connexion est sécurisée par un certificat, affiche ses données
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificat TLS :");
            Print("   Propriétaire :  ",subject);
            Print("   Emetteur :  ",issuer);
            Print("   Numéro :     ",serial);
            Print("   Print : ",thumbprint);
            Print("   Expiration : ",expiration);
            ExtTLS=true;
           }
         //--- envoi une requete GET au serveur
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Requête GET envoyée");
            //--- lit la réponse
            if(!HTTPRecv(socket,1000))
               Print("Echec d'obtention d'une réponse, erreur ",GetLastError());
           }
         else
            Print("Echec d'envoi de la requete GET, erreur ",GetLastError());
        }
      else
        {
         Print("Connexion à ",Address,":",Port," échouée, erreur ",GetLastError());
        }
      //--- ferme la socket après son utilisation
      SocketClose(socket);
     }
   else
      Print("Echec de création de la socket, erreur ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

SocketConnect

Connexion au serveur avec un contrôle du timeout.

```
bool  SocketConnect(
   int           socket,              // socket
   const string  server,              // adresse de connexion
   uint          port,                // port de connexion
   uint          timeout_receive_ms   // timeout de connexion
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est écrite dans _LastError.

server

[in] Nom de domaine du serveur auquel vous voulez vous connecter ou son adresse IP.

port

[in] Numéro du port.

timeout_receive_ms

[in] Timeout en millisecondes. Si la connexion n'est pas établie dans cet intervalle de temps, les tentatives sont stoppées.

Valeur de Retour

Si la connexion est réussie, retourne true, sinon false.

Note

L'adresse de connexion doit être ajoutée à la liste des adresses autorisées dans le terminal client (Outils \ Options \ Expert Advisors).

Si la connexion échoue, l'erreur 5272 (ERR_NETSOCKET_CANNOT_CONNECT) est écrite dans _LastError.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 - "Function is not allowed for call".

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Ajoute une adresse à la liste des adresses autorisées dans les
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Envoi une commande au serveur                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si la connexion TLS sécurisée est utilisée via le port 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si la connexion TCP standard est utilisée
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lit la réponse du serveur                                        |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lit les données des sockets tant qu'il y en a et juqu'au timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- différentes commandes de lecture suivant que la connexion est sécurisée
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analyse la réponse
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- n'affiche que l'en-tete de la réponse
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("En-tete de la réponse HTTP reçue :");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
```

```
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- vérifie le handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connexion si tout est ok
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connexion établie avec ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la connexion est sécurisée par un certificat, affiche ses données
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificat TLS :");
            Print("   Propriétaire :  ",subject);
            Print("   Emetteur :  ",issuer);
            Print("   Numéro :     ",serial);
            Print("   Print : ",thumbprint);
            Print("   Expiration : ",expiration);
            ExtTLS=true;
           }
         //--- envoi une requete GET au serveur
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Requête GET envoyée");
            //--- lit la réponse
            if(!HTTPRecv(socket,1000))
               Print("Echec d'obtention d'une réponse, erreur ",GetLastError());
           }
         else
            Print("Echec d'envoi de la requete GET, erreur ",GetLastError());
        }
      else
        {
         Print("Connexion à ",Address,":",Port," échouée, erreur ",GetLastError());
        }
      //--- ferme la socket après son utilisation
      SocketClose(socket);
     }
   else
      Print("Echec de création de la socket, erreur ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

SocketIsConnected

Vérifie si la socket est actuellement connectée.

```
bool  SocketIsConnected(
   const int  socket      // handle de la socket
   );
```

Parameters

socket

[in] Handle de la socket retourné par la fonction SocketCreate(). Lorsqu'un handle incorrect est passé à _LastError, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est activée.

Valeur de Retour

Retourne true si la socket est connectée, sinon - false.

Note

La fonction SocketIsConnected() permet de vérifier le statut actuel de connexion de la socket.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 – "Function is not allowed for call".

Voir également

SocketConnect, SocketIsWritable, SocketCreate, SocketClose

© 2000-2024, MetaQuotes Ltd.

SocketIsReadable

Retourne le nombre d'octets pouvant être lus sur une socket.

```
uint  SocketIsReadable(
   const int  socket      // handle de la socket
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé à _LastError, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est activée.

Valeur de Retour

Nombre d'octets pouvant être lus. Retourne 0 en cas d'erreur.

Note

Si une erreur se produit sur une socket système lors de l'exécution de la fonction, la connexion établie via SocketConnect est interrompue.

Avant d'appeler SocketRead, vérifiez si la socket comporte des données à lire. Sinon, s'il n'y a aucune donnée, la fonction SocketRead attends des données jusqu'à l'expiration de timeout_ms, retardant l'exécution du programme.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 – "Function is not allowed for call".

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Ajoute une adresse à la liste des adresses autorisées dans les
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Envoi une commande au serveur                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si la connexion TLS sécurisée est utilisée via le port 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si la connexion TCP standard est utilisée
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lit la réponse du serveur                                        |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lit les données des sockets tant qu'il y en a et juqu'au timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- différentes commandes de lecture suivant que la connexion est sécurisée
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analyse la réponse
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- n'affiche que l'en-tete de la réponse
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("En-tete de la réponse HTTP reçue :");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
```

```
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- vérifie le handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connexion si tout est ok
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connexion établie avec ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la connexion est sécurisée par un certificat, affiche ses données
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificat TLS :");
            Print("   Propriétaire :  ",subject);
            Print("   Emetteur :  ",issuer);
            Print("   Numéro :     ",serial);
            Print("   Print : ",thumbprint);
            Print("   Expiration : ",expiration);
            ExtTLS=true;
           }
         //--- envoi une requete GET au serveur
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Requête GET envoyée");
            //--- lit la réponse
            if(!HTTPRecv(socket,1000))
               Print("Echec d'obtention d'une réponse, erreur ",GetLastError());
           }
         else
            Print("Echec d'envoi de la requete GET, erreur ",GetLastError());
        }
      else
        {
         Print("Connexion à ",Address,":",Port," échouée, erreur ",GetLastError());
        }
      //--- ferme la socket après son utilisation
      SocketClose(socket);
     }
   else
      Print("Echec de création de la socket, erreur ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

SocketIsWritable

Vérifie si des données peuvent être écrites actuellement sur une socket.

```
bool  SocketIsWritable(
   const int  socket      // handle de la socket
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est écrite dans _LastError.

Valeur de Retour

Retourne true s'il est possible d'écrire, sinon false.

Note

Cette fonction vous permet de vérifier s'il est possible d'écrire actuellement des données sur une socket.

Si une erreur se produit sur une socket système lors de l'exécution de la fonction, la connexion établie via SocketConnect est interrompue.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 - "Function is not allowed for call".

© 2000-2024, MetaQuotes Ltd.

SocketTimeouts

Définit les timeouts pour recevoir et envoyer des données sur une socket système

```
bool  SocketTimeouts(
   int           socket,               // socket
   uint          timeout_send_ms,      // timeout d'envoi des données
   uint          timeout_receive_ms    // timeout d'obtention des données
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est écrite dans _LastError.

timeout_send_ms

[in] Timeout d'envoi des données en millisecondes.

timeout_receive_ms

[in] Timeout d'obtention des données en millisecondes.

Valeur de Retour

Retourne true en cas de succès, sinon false.

Note

Ne confondez pas les timeouts des objets systèmes avec ceux définis lors de la lecture des données via SocketRead. SocketTimeout définit les timeouts une seule fois pour une socket dans le système d'exploitation. Ces timeouts sont appliqués à toutes les fonctions pour la lecture et l'envoi des données via cette socket. Dans SocketRead, le timeout est défini pour une certaine opération de lecture des données.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 - "Function is not allowed for call".

© 2000-2024, MetaQuotes Ltd.

SocketRead

Lit les données d'une socket.

```
int  SocketRead(
   int           socket,              // socket
   uchar&        buffer[],            // buffer pour lire les données depuis une soc
   uint          buffer_maxlen,       // nombre d'octets à lire
   uint          timeout_ms           // timeout
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé à _LastError, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est activée.

buffer

[out] Référence au tableau de type uchar dans lequel les données sont lues. La taille du tableau dynamique est augmentée du nombre d'octets lus. La taille du tableau ne peut pas être supérieure à INT_MAX (2147483647).

buffer_maxlen

[in] Nombre d'octets à lire dans le tableau *buffer[]*. Les données ne rentrant pas dans le tableau restent dans la socket. Elles peuvent être lues lors de l'appel suivant à SocketRead. *buffer_maxlen* ne peut pas être supérieur à INT_MAX (2147483647).

timeout_ms

[in] Timeout de lecture des données en millisecondes. Si les données ne sont pas obtenues dans le temps imparti, les tentatives sont stoppées et la fonction retourne -1.

Valeur de Retour

Retourne le nombre d'octets lus en cas de succès. Retourne -1 en cas d'erreur.

Note

Si une erreur se produit sur une socket système lors de l'exécution de la fonction, la connexion établie via SocketConnect est interrompue.

En cas d'erreur de lecture des données, l'erreur 5273 (ERR_NETSOCKET_IO_ERROR) est inscrite dans _LastError.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 – "Function is not allowed for call".

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Ajoutez une adresse à la liste des adresses autorisées dans les
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Envoie la commande au serveur                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si la connexion sécurisée TLS est utilisée via le port 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si une connexion TCP standard est utilisée
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lit la réponse du serveur                                        |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lit les données des sockets tant qu'il y en a, et jusqu'au timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- différentes commande de lecture suivant que la connexion est sécurisée
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analyse la réponse
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- n'affiche que l'en-tete de la réponse
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("En-tete de la réponse HTP reçue :");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
```

```
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- vérifie le handle
   if(socket!=INVALID_HANDLE)
     {
      //--- établit la connexion si tout est ok
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connexion établie avec ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la connexion est sécurisée par un certificat, affiche ses données
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificat TLS :");
            Print("   Propriétaire :  ",subject);
            Print("   Emetteur :  ",issuer);
            Print("   Numéro :     ",serial);
            Print("   Print : ",thumbprint);
            Print("   Expiration : ",expiration);
            ExtTLS=true;
           }
         //--- envoie une requete GET au serveur
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Requête GET envoyée");
            //--- lit la réponse
            if(!HTTPRecv(socket,1000))
               Print("Echec d'obtention d'une réponse, erreur ",GetLastError());
           }
         else
            Print("Echec d'envoi de la requete GET, erreur ",GetLastError());
        }
      else
        {
         Print("Echec de la connexion à ",Address,":",Port,", erreur ",GetLastError())
        }
      //--- ferme la socket après son utilisation
      SocketClose(socket);
     }
   else
      Print("Echec de création d'une socket, erreur ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Voir également

SocketTimeouts, MathSwap

© 2000-2024, MetaQuotes Ltd.

SocketSend

Ecrit des données sur une socket.

```
int  SocketSend(
   int           socket,              // socket
   const uchar&  buffer[],            // buffer de données
   uint          buffer_len           // taille du buffer
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est écrite dans _LastError.

buffer

[in] Référence à un tableau de type uchar contenant les données à envoyer sur la socket.

buffer_len

[in] Taille du tableau 'buffer'.

Valeur de Retour

En cas de succès, retourne le nombre d'octets écrits sur une socket. Retourne -1 en cas d'erreur.

Note

Si une erreur se produit sur une socket système lors de l'exécution de la fonction, la connexion établie via SocketConnect est interrompue.

En cas d'erreur d'écriture des données, l'erreur 5273 (ERR_NETSOCKET_IO_ERROR) est inscrite dans _LastError.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 – "Function is not allowed for call".

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Ajoute une adresse à la liste des adresses autorisées dans les
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Envoi une commande au serveur                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si la connexion TLS sécurisée est utilisée via le port 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si la connexion TCP standard est utilisée
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lit la réponse du serveur                                        |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lit les données des sockets tant qu'il y en a et juqu'au timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- différentes commandes de lecture suivant que la connexion est sécurisée
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analyse la réponse
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- n'affiche que l'en-tete de la réponse
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("En-tete de la réponse HTTP reçue :");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
```

```
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- vérifie le handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connexion si tout est ok
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connexion établie avec ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la connexion est sécurisée par un certificat, affiche ses données
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificat TLS :");
            Print("   Propriétaire :  ",subject);
            Print("   Emetteur :  ",issuer);
            Print("   Numéro :     ",serial);
            Print("   Print : ",thumbprint);
            Print("   Expiration : ",expiration);
            ExtTLS=true;
           }
         //--- envoi une requete GET au serveur
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Requête GET envoyée");
            //--- lit la réponse
            if(!HTTPRecv(socket,1000))
               Print("Echec d'obtention d'une réponse, erreur ",GetLastError());
           }
         else
            Print("Echec d'envoi de la requete GET, erreur ",GetLastError());
        }
      else
        {
         Print("Connexion à ",Address,":",Port," échouée, erreur ",GetLastError());
        }
      //--- ferme la socket après son utilisation
      SocketClose(socket);
     }
   else
      Print("Echec de création de la socket, erreur ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Voir également

SocketTimeouts, MathSwap, StringToCharArray

© 2000-2024, MetaQuotes Ltd.

SocketTlsHandshake

Initie une connexion TLS (SSL) vers un hôte spécifié via le protocole TLS Handshake. Pendant ce Handshake, un client et un serveur se mettent d'accord sur les paramètres de connexion : version du protocole appliqué et méthode d'encryptage des données.

```
bool  SocketTlsHandshake(
   int           socket,                // socket
   const string  host                   // adresse de l'hôte
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est écrite dans _LastError.

host

[in] Adresse d'un hôte avec lequel la conexion sécurisée est établie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

Notes

Avant une connexion sécurisée, le programle doit établir une connexion TCP standard avec l'hôte en utilisant SocketConnect.

Si la connexion sécurisée échoue, l'erreur 5274 (ERR_NETSOCKET_HANDSHAKE_FAILED) est écrite dans _LastError.

Il n'y a pas besoin d'appeler la fonction lors d'une connexion au port 443. C'est un port TCP standard utilisé pour les connexions sécurisées TLS (SSL).

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 – "Function is not allowed for call".

© 2000-2024, MetaQuotes Ltd.

SocketTlsCertificate

Retourne les données sur le certificat utilisé pour une connexion sécurisée au réseau

```
int  SocketTlsCertificate(
   int           socket,              // socket
   string&       subject,             // propriétaire du certificat
   string&       issuer,              // émetteur du certificat
   string&       serial,              // numéro de série du certificat
   string&       thumbprint,          // print du certificat
   datetime&     expiration           // expiration du certificat
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est écrite dans _LastError.

subject

[in] Nom du propriétaire du certificat. Correspond au champ Sujet.

issuer

[in] Nom de l'émetteur du certificat. Correspond au champ Emetteur.

serial

[in] Numéro de série du certificat. Correspond au champ SerialNumber.

thumbprint

[in] Print du certificat. Correspond au numéro de hachage SHA-1 du fichier du certificat entier (tous les champs incluant la signature de l'émetteur).

expiration

[in] Date d'expiration du certificat au format datetime.

Valeur de Retour

Retourne true en cas de succès, sinon false.

Note

Les données du certificat ne peuvent être obtenues qu'après avoir établi une connexion sécurisée avec SocketTlsHandshake.

En cas d'erreur d'obtention d'un certificat, l'erreur 5275 (ERR_NETSOCKET_NO_CERTIFICATE) est écrite dans _LastError.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 – "Function is not allowed for call".

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Ajoute une adresse à la liste des adresses autorisées dans les
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Envoi une commande au serveur                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si la connexion TLS sécurisée est utilisée via le port 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si la connexion TCP standard est utilisée
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lit la réponse du serveur                                        |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lit les données des sockets tant qu'il y en a et juqu'au timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- différentes commandes de lecture suivant que la connexion est sécurisée
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analyse la réponse
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- n'affiche que l'en-tete de la réponse
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("En-tete de la réponse HTTP reçue :");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
```

```
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- vérifie le handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connexion si tout est ok
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connexion établie avec ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la connexion est sécurisée par un certificat, affiche ses données
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificat TLS :");
            Print("   Propriétaire :  ",subject);
            Print("   Emetteur :  ",issuer);
            Print("   Numéro :     ",serial);
            Print("   Print : ",thumbprint);
            Print("   Expiration : ",expiration);
            ExtTLS=true;
           }
         //--- envoi une requete GET au serveur
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Requête GET envoyée");
            //--- lit la réponse
            if(!HTTPRecv(socket,1000))
               Print("Echec d'obtention d'une réponse, erreur ",GetLastError());
           }
         else
            Print("Echec d'envoi de la requete GET, erreur ",GetLastError());
        }
      else
        {
         Print("Connexion à ",Address,":",Port," échouée, erreur ",GetLastError());
        }
      //--- ferme la socket après son utilisation
      SocketClose(socket);
     }
   else
      Print("Echec de création de la socket, erreur ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

SocketTlsRead

Lit des données depuis une connexion TLS sécurisée.

```
int  SocketTlsRead(
   int           socket,              // socket
   uchar&        buffer[],            // buffer pour lire les données depuis une soc
   uint          buffer_maxlen        // nombre d'octets à lire
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé à _LastError, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est activée.

buffer

[out] Référence au tableau de type uchar dans lequel les données sont lues. La taille du tableau dynamique est augmentée du nombre d'octets lus. La taille du tableau ne peut pas être supérieure à INT_MAX (2147483647).

buffer_maxlen

[in] Nombre d'octets à lire dans le tableau *buffer[]*. Les données ne rentrant pas dans le tableau restent dans la socket. Elles peuvent être lues lors de l'appel suivant à SocketTLSRead. *buffer_maxlen* ne peut pas être supérieur à INT_MAX (2147483647).

Valeur de Retour

Retourne le nombre d'octets lus en cas de succès. Retourne -1 en cas d'erreur.

Note

Si une erreur se produit sur une socket système lors de l'exécution de la fonction, la connexion établie via SocketConnect est interrompue.

La fonction est exécutée jusqu'à la réception de la quantité spécifiée de données, ou si le timeout est atteint (SocketTimeouts).

En cas d'erreur de lecture des données, l'erreur 5273 (ERR_NETSOCKET_IO_ERROR) est inscrite dans _LastError.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 – "Function is not allowed for call".

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Ajoutez une adresse à la liste des adresses autorisées dans les
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Envoie la commande au serveur                                    |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- si la connexion sécurisée TLS est utilisée via le port 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- si une connexion TCP standard est utilisée
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lit la réponse du serveur                                        |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lit les données des sockets tant qu'il y en a, et jusqu'au timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- différentes commande de lecture suivant que la connexion est sécurisée
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analyse la réponse
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- n'affiche que l'en-tete de la réponse
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("En-tete de la réponse HTP reçue :");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
```

```
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- vérifie le handle
   if(socket!=INVALID_HANDLE)
     {
      //--- établit la connexion si tout est ok
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connexion établie avec ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- si la connexion est sécurisée par un certificat, affiche ses données
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificat TLS :");
            Print("   Propriétaire :  ",subject);
            Print("   Emetteur :  ",issuer);
            Print("   Numéro :     ",serial);
            Print("   Print : ",thumbprint);
            Print("   Expiration : ",expiration);
            ExtTLS=true;
           }
         //--- envoie une requete GET au serveur
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Requête GET envoyée");
            //--- lit la réponse
            if(!HTTPRecv(socket,1000))
               Print("Echec d'obtention d'une réponse, erreur ",GetLastError());
           }
         else
            Print("Echec d'envoi de la requete GET, erreur ",GetLastError());
        }
      else
        {
         Print("Echec de la connexion à ",Address,":",Port,", erreur ",GetLastError())
        }
      //--- ferme la socket après son utilisation
      SocketClose(socket);
     }
   else
      Print("Echec de création d'une socket, erreur ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Voir également

SocketTimeouts, MathSwap

© 2000-2024, MetaQuotes Ltd.

SocketTlsReadAvailable

Lit toutes les données disponibles depuis une connexion TLS sécurisée.

```
int  SocketTlsReadAvailable(
   int           socket,              // socket
   uchar&        buffer[],            // buffer pour lire les données depuis une soc
   const uint    buffer_maxlen        // nombre d'octets à lire
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé à _LastError, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est activée.

buffer

[out] Référence au tableau de type uchar dans lequel les données sont lues. La taille du tableau dynamique est augmentée du nombre d'octets lus. La taille du tableau ne peut pas être supérieure à INT_MAX (2147483647).

buffer_maxlen

[in] Nombre d'octets à lire dans le tableau buffer[]. Les données ne rentrant pas dans le tableau restent dans la socket. Elles peuvent être lues lors de l'appel suivant à SocketTlsReadAvailable ou à SocketTlsRead. buffer_maxlen ne peut pas être supérieur à INT_MAX (2147483647).

Valeur de Retour

Retourne le nombre d'octets lus en cas de succès. Retourne -1 en cas d'erreur.

Note

Si une erreur se produit sur une socket système lors de l'exécution de la fonction, la connexion établie via SocketConnect est interrompue.

En cas d'erreur de lecture des données, l'erreur 5273 (ERR_NETSOCKET_IO_ERROR) est inscrite dans _LastError.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 – "Function is not allowed for call".

Voir également

SocketTimeouts, MathSwap

© 2000-2024, MetaQuotes Ltd.

SocketTlsSend

Envoi des données via une connexion TLS sécurisée.

```
int  SocketTlsSend(
   int           socket,              // socket
   const uchar&  buffer[],            // buffer de données
   uint          buffer_len           // taille du buffer
   );
```

Parameters

socket

[in] Handle de la socket retournée par la fonction SocketCreate. Lorsqu'un handle incorrect est passé, l'erreur 5270 (ERR_NETSOCKET_INVALIDHANDLE) est écrite dans _LastError.

buffer

[in] Référence à un tableau de type uchar contenant les données à envoyer.

buffer_len

[in] Taille du tableau 'buffer'.

Valeur de Retour

En cas de succès, retourne le nombre d'octets écrits sur une socket. Retourne -1 en cas d'erreur.

Note

Si une erreur se produit sur une socket système lors de l'exécution de la fonction, la connexion établie via SocketConnect est interrompue.

En cas d'erreur d'écriture des données, l'erreur 5273 (ERR_NETSOCKET_IO_ERROR) est inscrite dans _LastError.

La fonction ne peut être appelée que depuis les Expert Advisors et les scripts, puisqu'ils sont exécutés dans leurs propres threads d'exécution. Si appelé depuis un indicateur, GetLastError() retourne l'erreur 4014 – "Function is not allowed for call".

Voir également

SocketTimeouts, MathSwap, StringToCharArray

© 2000-2024, MetaQuotes Ltd.

WebRequest

The function sends an HTTP request to a specified server. The function has two versions:

1. **Sending simple requests** of type "key=value" using the header Content-Type: application/x-www-form-urlencoded.

```
int  WebRequest(
   const string      method,           // HTTP method
   const string      url,              // URL
   const string      cookie,           // cookie
   const string      referer,          // referer
   int               timeout,          // timeout
   const char        &data[],          // the array of the HTTP message body
   int               data_size,        // data[] array size in bytes
   char              &result[],        // an array containing server response data
   string            &result_headers   // headers of server response
   );
```

2. **Sending a request of any type** specifying the custom set of headers for a more flexible interaction with various Web services.

```
int  WebRequest(
   const string      method,           // HTTP method
   const string      url,              // URL
   const string      headers,          // headers
   int               timeout,          // timeout
   const char        &data[],          // the array of the HTTP message body
   char              &result[],        // an array containing server response data
   string            &result_headers   // headers of server response
   );
```

Parameters

method

[in] HTTP method.

url

[in] URL.

headers

[in] Request headers of type "key: value", separated by a line break "\r\n".

cookie

[in] Cookie value.

referer

[in] Value of the Referer header of the HTTP request.

timeout

[in] Timeout in milliseconds.

© 2000-2024, MetaQuotes Ltd.

data[]

[in] Data array of the HTTP message body.

data_size

[in] Size of the data[] array.

result[]

[out] An array containing server response data.

result_headers

[out] Server response headers.

Return Value

HTTP server response code or -1 for an error.

Note

To use the WebRequest() function, add the addresses of the required servers in the list of allowed URLs in the "Expert Advisors" tab of the "Options" window. Server port is automatically selected on the basis of the specified protocol - 80 for "http://" and 443 for "https://".

The WebRequest() function is synchronous, which means its breaks the program execution and waits for the response from the requested server. Since the delays in receiving a response can be large, the function is not available for calls from the indicators, because indicators run in a common thread shared by all indicators and charts on one symbol. Indicator performance delay on one of the charts of a symbol may stop updating of all charts of the same symbol.

The function can be called only from Expert Advisors and scripts, as they run in their own execution threads. If you try to call the function from an indicator, GetLastError() will return error 4014 - "Function is not allowed for call".

WebRequest() cannot be executed in the Strategy Tester.

Exemple :

```
void OnStart()
  {
   string cookie=NULL,headers;
   char   post[],result[];
   string url="https://finance.yahoo.com";
//--- To enable access to the server, you should add URL "https://finance.yahoo.com"
//--- to the list of allowed URLs (Main Menu->Tools->Options, tab "Expert Advisors"):
//--- Resetting the last error code
   ResetLastError();
//--- Downloading a html page from Yahoo Finance
   int res=WebRequest("GET",url,cookie,NULL,500,post,0,result,headers);
   if(res==-1)
     {
      Print("Error in WebRequest. Error code  =",GetLastError());
      //--- Perhaps the URL is not listed, display a message about the necessity to ad
      MessageBox("Add the address '"+url+"' to the list of allowed URLs on tab 'Expert
     }
```

© 2000-2024, MetaQuotes Ltd.

```
  else
    {
     if(res==200)
       {
        //--- Successful download
        PrintFormat("The file has been successfully downloaded, File size %d byte.",A
        //PrintFormat("Server headers: %s",headers);
        //--- Saving the data to a file
        int filehandle=FileOpen("url.htm",FILE_WRITE|FILE_BIN);
        if(filehandle!=INVALID_HANDLE)
          {
           //--- Saving the contents of the result[] array to a file
           FileWriteArray(filehandle,result,0,ArraySize(result));
           //--- Closing the file
           FileClose(filehandle);
          }
        else
           Print("Error in FileOpen. Error code =",GetLastError());
       }
     else
        PrintFormat("Downloading '%s' failed, error code %d",url,res);
    }
}
```

© 2000-2024, MetaQuotes Ltd.

SendFTP

Envoie le fichier à l'adresse indiquée dans la fenêtre des réglages du signet "la publication".

```
bool  SendFTP(
   string  filename,          // fichier pour l'envoi à
   string  ftp_path=NULL      // le chemin vers le déchargement sur le serveur ftp
   );
```

Paramètres

filename

[in] Le nom du fichier envoyé.

ftp_path=NULL

[in] Le répertoire FTP. Si le répertoire n'est pas indiqué, on utilise le répertoire décrit dans les configurations.

La valeur rendue

En cas de l'échec rend false.

Note

Le fichier envoyé doit se trouver dans le dossier *le répertoire_du terminal\MQL5\files* ou dans ses sous-dossiers. L'envoi n'est pas produit, si dans les configurations on n'indique pas l'adresse FTP et/ou le mot de passe de l'accès.

При работе в тестере стратегий функция SendFTP() не выполняется.

© 2000-2024, MetaQuotes Ltd.

SendMail

Envoie le message électronique à l'adresse indiquée dans la fenêtre des réglages du signet "la Poste"

```
bool  SendMail(
   string  subject,        // titre
   string  some_text       // texte du message
   );
```

Paramètres

subject

[in] Le titre du message.

some_text

[in] Le corps du message.

La valeur rendue

true - si le message est mise au tour sur l'envoi, autrement rend false.

Note

L'envoi peut être interdit dans les configurations, ou peut être on n'a pas indiqué l'adresse de courrier électronique. Pour recevoir l'information sur l'erreur, il est nécessaire d' appeler la fonction GetLastError().

При работе в тестере стратегий функция SendMail() не выполняется.

© 2000-2024, MetaQuotes Ltd.

SendNotification

Envoie la notification aux terminaux mobiles dont ID MetaQuotes sont spécifiés dans la fenêtre des options sur l'onglet "Notifications"

```
bool  SendNotification(
   string  text          // le texte du message
   );
```

Paramètres

text

[in] Le texte du message dans la notification. La longueur du message doit être pas plus de 255 symboles.

La valeur rendue

true) l'envoi réussi de la notification à partir du terminal, en cas de l'échec rend false. A la vérification après un mauvais envoi de la notificationGetLastError() une des erreurs suivante peut être donnée:

- 4515 – ERR_NOTIFICATION_SEND_FAILED,
- 4516 – ERR_NOTIFICATION_WRONG_PARAMETER,
- 4517 – ERR_NOTIFICATION_WRONG_SETTINGS,
- 4518 – ERR_NOTIFICATION_TOO_FREQUENT.

Note

Pour la fonction SendNotification() il y a des limitations strictes selon l'utilisation: pas plus de 2 appels par seconde et pas plus de 10 appels par minute. Le contrôle de la fréquence de l'utilisation est fait dynamiquement, et la fonction peut être bloquée à la violation.

При работе в тестере стратегий функция SendNotification() не выполняется.

© 2000-2024, MetaQuotes Ltd.

Les variables globales du terminal de client

Le groupe des fonctions destinées au travail avec les variables globales.

Il ne faut pas confondre les variables globales du terminal de client avec les variables déclarées au niveau global du programme mql5.

Les variables globales existent dans le terminal de client 4 semaines dès le moment du dernier appel, après cela elles sont supprimés automatiquement. Non seulement l'installation de la nouvelle valeur est considérée comme l'appel vers la variable globale, mais aussi la lecture de la valeur de la variable globale.

Les variables globales du terminal de client sont accessibles simultanément de tous les programmes mql5 lancés sur le terminal de client.

Fonction	Action
GlobalVariableCheck	Contrôle l'existence de la variable globale avec le nom indiqué
GlobalVariableTime	Rend le temps du dernier accès à la variable globale
GlobalVariableDel	Supprime la variable globale
GlobalVariableGet	Demande la valeur de la variable globale
GlobalVariableName	Rend le nom de la variable globale selon le numéro d'ordre dans le répertoire des variables globales
GlobalVariableSet	Met la nouvelle valeur à une variable globale
GlobalVariablesFlush	Inscrit forcément le contenu de toutes les variables globales sur le disque
GlobalVariableTemp	Met une nouvelle valeur de la variable globale, qui existe seulement pendant la session courante du terminal
GlobalVariableSetOnCondition	Met une nouvelle valeur de la variable existant globale selon la condition
GlobalVariablesDeleteAll	Supprime les variables globales avec le préfixe indiqué dans le nom
GlobalVariablesTotal	Rend le total des variables globales

© 2000-2024, MetaQuotes Ltd.

GlobalVariableCheck

Contrôle l'existence de la variable globale avec le nom indiqué.

```
bool  GlobalVariableCheck(
   string  name      // nom
   );
```

Paramètres

name

[in] Le nom de la variable globale.

La valeur rendue

Rend la valeur true, si la variable globale existe, autrement rend false.

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

Voir aussi

GlobalVariableTime()

© 2000-2024, MetaQuotes Ltd.

GlobalVariableTime

Rend le temps du dernier accès à la variable globale.

```
datetime  GlobalVariableTime(
   string  name      // nom
   );
```

Paramètres

name

[in] Le nom de la variable globale.

La valeur rendue

Rend le temps du dernier accès à la variable indiquée globale. L'appel à la variable pour la valeur est aussi l'accès. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

Voir aussi

GlobalVariableCheck()

© 2000-2024, MetaQuotes Ltd.

GlobalVariableDel

Supprime la variable globale du terminal de client.

```
bool  GlobalVariableDel(
   string  name      // nom
   );
```

Paramètres

name

[in] Le nom de la variable globale.

La valeur rendue

A l'effacement réussi la fonction rend true, autrement rend false. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

© 2000-2024, MetaQuotes Ltd.

GlobalVariableGet

Rend la valeur de la variable globale existante du terminal de client. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
double  GlobalVariableGet(
   string  name      // nom
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la variable globale du terminal de client se place à la variable de réception transmise selon le lien par le deuxième paramètre.

```
bool  GlobalVariableGet(
   string  name               // nom
   double& double_var         // acceptons ici la valeur de la variable globale
   );
```

Paramètres

name

[in] Le nom de la variable globale.

double_var

[out] La variable du type double, acceptant la valeur qui se trouve dans la variable globale du terminal de client.

La valeur rendue

La valeur de la variable globale existante ou 0 en cas de l'erreur. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

© 2000-2024, MetaQuotes Ltd.

GlobalVariableName

Rend le nom de la variable globale selon le numéro d'ordre.

```
string  GlobalVariableName(
   int  index      // numéro dans le répertoire des variables globales
   );
```

Paramètres

index

[in] Le numéro d'ordre dans le répertoire des variables globales. Doit être plus ou égal à 0 et moins que GlobalVariablesTotal().

La valeur rendue

Le nom de la variable globale selon le numéro d'ordre dans la liste des variables globales.Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

© 2000-2024, MetaQuotes Ltd.

GlobalVariableSet

Met la nouvelle valeur à une variable globale. Si la variable n'existe pas, le système crée une nouvelle variable globale.

```
datetime  GlobalVariableSet(
   string  name,       // nom
   double  value       // valeur à mettre
   );
```

Paramètres

name

[in] Le nom de la variable globale.

value

[in] Une nouvelle valeur numérique.

La valeur rendue

À l'exécution réussie la fonction rend le temps du dernier accès, autrement c'est 0. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Имя графического объекта не должно превышать 63 символа. Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

© 2000-2024, MetaQuotes Ltd.

GlobalVariablesFlush

Inscrit forcément le contenu de toutes les variables globales sur le disque.

```
void  GlobalVariablesFlush();
```

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Le terminal inscrit lui-même toutes les variables globales à la fin du travail, mais à la panne soudaine du travail de l'ordinateur les données peuvent se perdre. Cette fonction permet de maîtriser indépendamment le procès de la sauvegarde des variables globales sur le cas des situations anormales.

© 2000-2024, MetaQuotes Ltd.

GlobalVariableTemp

Produit la tentative de la création de la variable temporaire globale. Si la variable n'existe pas, le système crée une nouvelle variable temporaire globale.

```
bool  GlobalVariableTemp(
   string  name,       // nom
   );
```

Paramètres

name

[in] Le nom de la variable temporaire globale.

La valeur rendue

A l'exécution réussi la fonction rend true, autrement rend false. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Les variables temporaires globales existent seulement au temps de travail du terminal de client, après la clôture du terminal ils sont supprimés automatiquement. A l'exécution de l'opération GlobalVariablesFlush() les variables temporaires globales ne s'inscrivent pas sur le disque.

Après la création de la variable temporaire globale l'accès à cette variable et sa modification se réalise ainsi que vers la variable ordinaire globale du terminal de client.

© 2000-2024, MetaQuotes Ltd.

GlobalVariableSetOnCondition

Met une nouvelle valeur de la variable globale existante si la valeur courante est égal à la valeur du troisième paramètre check_value. Si la variable n'existe pas, la fonction générera l'erreur ERR_GLOBALVARIABLE_NOT_FOUND (4501) et rendra false.

```
bool  GlobalVariableSetOnCondition(
   string  name,            // nom
   double  value,           // valeur à l'exécution de la condition
   double  check_value      // condition vérifiée
   );
```

Paramètres

name

[in] Le nom de la variable globale.

value

[in] Une nouvelle valeur.

check_value

[in] La valeur pour la vérification de la valeur courante de la variable globale.

La valeur rendue

A l'exécution réussi la fonction rend true, autrement rend false. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError(). Si la valeur courante de la variable globale se distingue de check_value, la fonction rendra false.

Note

La fonction assure l'accès atomique à la variable globale, c'est pourquoi elle peut être utilisée pour l'organisation du mutex à la coopération de quelques experts travaillant simultanément dans la limite d'un terminal de client.

© 2000-2024, MetaQuotes Ltd.

GlobalVariablesDeleteAll

Supprime les variables globales du terminal de client.

```
int  GlobalVariablesDeleteAll(
   string     prefix_name=NULL      // toutes les variables globales, dont les noms co
   datetime   limit_data=0          // toutes les variables globales, qui changeaient
   );
```

Paramètres

prefix_name=NULL

[in] Le préfixe du nom des variables globales supprimées. Si on indique le préfixe NULL ou la chaîne vide, toutes les variables globales conviennent au critère de l'effacement ,correspondantes au critère de l'effacement selon la date

limit_data=0

[in] La date pour la sélection des variables globales par le temps de la dernière modification. Les variables globales se suppriment, qui changeaient avant la date indiquée. Si le paramètre est égal au zéro, toutes les variables globales, correspondantes au premier critère (selon le préfixe) se suppriment.

La valeur rendue

Le nombre de variables supprimées.

Note

Si les deux paramètres sont égaux au zéro (prefix_name=NULL et limit_data=0),toutes les variables globales du terminal se suppriment. Si on indique les deux paramètres, se suppriment les variables globales correspondantes simultanément à chacun des paramètres indiqués.

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

© 2000-2024, MetaQuotes Ltd.

GlobalVariablesTotal

Rend le total des variables globales du terminal de client.

```
int  GlobalVariablesTotal();
```

La valeur rendue

Le nombre de variables globales.

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement. Non seulement l'installation de la nouvelle valeur est considérée comme l'appel à la variable globale, mais aussi la lecture de la valeur de la variable globale.

© 2000-2024, MetaQuotes Ltd.

Les opérations de fichier

Le groupe des fonctions pour le travail avec les fichiers.

Il y a deux répertoires (avec les sous-directoires), dans lesquels les fichiers de travail peuvent être installés:

- le répertoire_des données_du terminal\MQL5\FILES\ (choisissez le point du menu "Fichier" - "Ouvrir le répertoire des données" pour voir dans le terminal);
- le dossier total de tous les terminaux établis sur l'ordinateur – est d'habitude disposé dans le répertoire C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal\Common\Files.

D'une manière de programme on peut recevoir les noms de ces répertoires à l'aide de la fonction TerminalInfoString(), en utilisant les énumérations ENUM_TERMINAL_INFO_STRING:

```
//--- dossier, où se trouvent les données du terminal
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- Dossier total de tous les terminaux de client
   string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

Le travail avec les fichiers d'autres répertoires est interdit.

Les fonctions de fichier permettent de travailler avec des soi-disant "canaux nommés". Pour cela il suffit d'appeler la fonctionFileOpen() avec des paramètres appropriés.

Fonction	Action
FileSelectDialog	Crée une boîte de dialogue d'ouverture/création d'un fichier ou d'un répertoire
FileFindFirst	Commence l'excédent des fichiers dans le répertoire correspondant conformément au filtre indiqué
FileFindNext	Continue la recherche commencée par la fonction FileFindFirst()
FileFindClose	Ferme le handle de la recherche
FileOpen	Ouvre un fichier avec un nom et un drapeau indiqués
FileDelete	Supprime le fichier indiqué
FileFlush	Écrit à un disque toutes les données restant dans le tampon du fichier d'entrée-de sortie
FileGetInteger	Reçoit la propriété entière du fichier
FileIsEnding	Définit la fin du fichier en train de la lecture
FileIsLineEnding	Définit la fin de la ligne dans le fichier du texte en train de la lecture
FileClose	Ferme le fichier auparavant ouvert
FileIsExist	Vérifie l'existence d'un fichier

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
FileCopy	Copie le fichier original d'un répertoire local ou commun à un autre fichier
FileMove	Déplace et renomme le fichier
FileReadArray	Lit les tableaux de tous les types, sauf les tableaux de chaîne (peut être un tableau des structures qui ne contient pas les chaînes et les tableaux dynamiques), du fichier binaire de la position courante de l'indicateur de fichier
FileReadBool	Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type bool
FileReadDatetime	Lit du fichier du type CSV la chaîne d'un des formats: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" ou "HH:MI:SS" - et la transforme en valeur du type datetime
FileReadDouble	Lit le nombre de l'exactitude double avec une virgule flottante (double) du fichier binaire de la position courante de l'indicateur de fichier
FileReadFloat	Lit de la position courante de l'indicateur de fichier la valeur du type float
FileReadInteger	Lit du fichier binaire la valeur du type int, short ou char en fonction de la longueur indiquée dans les bytes
FileReadLong	Lit de la position courante de l'indicateur de fichier la valeur du type long
FileReadNumber	Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type double
FileReadString	Lit du fichier la chaîne de la position courante de l'indicateur de fichier
FileReadStruct	Lit du fichier binaire le contenu à la structure transmise à titre du paramètre
FileSeek	Déplace la position de l'indicateur de fichier par un nombre spécifié de bytes par rapport à la position indiquée
FileSize	Rend la dimension d'un fichier ouvert correspondant
FileTell	Rend la position courante de l'indicateur de fichier du fichier correspondant ouvert
FileWrite	Inscrit les données au fichier comme CSV ou TXT
FileWriteArray	Inscrit au fichier du type BIN les tableaux des tous les types, sauf les tableaux de chaîne

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
FileWriteDouble	Inscrit au fichier binaire la valeur du paramètre du type double de la position courante de l'indicateur de fichier
FileWriteFloat	Inscrit au fichier binaire la valeur du paramètre du type float de la position courante de l'indicateur de fichier
FileWriteInteger	Inscrit au fichier binaire la valeur du paramètre du type int de la position courante de l'indicateur de fichier
FileWriteLong	Inscrit au fichier binaire la valeur du paramètre du type long de la position courante de l'indicateur de fichier
FileWriteString	Inscrit au fichier du type BIN ou TXT la valeur du paramètre du type string de la position courante de l'indicateur de fichier
FileWriteStruct	Inscrit au fichier binaire le contenu de la structure, transmise à titre du paramètre de la position courante de l'indicateur de fichier
FileLoad	Lit toutes les données d'un fichier binaire spécifié et remplit le tableau d'éléments numériques ou de structures simples passé en argument
FileSave	Ecrit tous les éléments d'un tableau passé en argument dans un fichier binaire
FolderCreate	Crée le répertoire dans le dossier Files (en fonction de la valeur common_flag)
FolderDelete	Supprime le répertoire indiqué. Si le dossier n'est pas vide, elle ne peut pas être supprimé
FolderClean	Supprime tous les fichiers dans le dossier indiqué

Si le fichier s'ouvre pour l'enregistrement à l'aide de la fonction FileOpen(), tous les sous-dossiers, indiqué au chemin seront créés en cas de leur absence.

© 2000-2024, MetaQuotes Ltd.

FileSelectDialog

Crée une boîte de dialogue d'ouverture/création d'un fichier ou d'un répertoire.

```
int  FileSelectDialog(
   string   caption,              // en-tête de la fenêtre
   string   initial_dir,          // répertoire initial
   string   filter,               // filtre d'extensions
   uint     flags,                // combinaison de flags
   string&  filenames[],          // tableau avec les noms des fichiers
   string   default_filename      // nom de fichier par défaut
   );
```

Paramètres

caption

[in] En-tête de la boîte de dialogue.

initial_dir

[in] Nom du répertoire initial relativement à MQL5\Files, dont le contenu est affiché dans la boîte de dialogue. Si la valeur est NULL, le répertoire MQL5\Files est affiché.

filter

[in] Filtre d'extensions des fichiers à afficher dans la boîte de sélection. Les fichiers ayant un autre format sont cachés.

flags

[in] Combinaison de flags définissant le mode de la fenêtre de dialogue. Les flags sont définis comme suit :
FSD_WRITE_FILE - boîte de dialogue d'ouverture de fichier ;
FSD_SELECT_FOLDER - permet la sélection des répertoires uniquement ;
FSD_ALLOW_MULTISELECT - permet la sélection de plusieurs fichiers ;
FSD_FILE_MUST_EXIST - les fichiers sélectionnés doivent exister ;
FSD_COMMON_FOLDER - le fichier est situé dans le répertoire commun à tous les terminaux clients \Terminal\Common\Files.

filenames[]

[out] Tableau de chaînes de caractères dans lequel sont placés les fichiers/répertoires sélectionnés.

default_filename

[in] Nom du fichier/répertoire par défaut. S'il est spécifié, le nom est automatiquement ajouté à la boîte de dialogue et est retourné dans le tableau *filenames[]* lors des tests.

Valeur de Retour

En cas de succès, la fonction retourne le nombre de fichiers sélectionnés dont les noms sont dans *filenames[]*. Si l'utilisateur ferme la boîte de dialogue dans avoir sélectionné de fichier, la fonction retourne 0. En cas d'exécution infructueuse, une valeur inférieure à 0 est retournée. Le code d'erreur peut être obtenu avec GetLastError().

© 2000-2024, MetaQuotes Ltd.

Note

Pour des raisons de sécurité, l'utilisation de fichiers est strictement contrôlée dans le langage MQL5. Les fichiers utilisés dans les opérations sur les fichiers avec le langage MQL5 ne peuvent pas être situés en dehors du bac à sable (c'est à dire en dehors du répertoire MQL5\Files).

Le nom *initial_dir* est recherché dans le répertoire du terminal client dans MQL5\Files (ou répertoire_agent_de_test\MQL5\Files en cas de test). Si FSD_COMMON_FOLDER est utilisé dans les flags, la recherche du répertoire initial est effectuée dans le répertoire commun à tous les terminaux clients \Terminal\Common\Files.

Le paramètre *filter* indique les fichiers valides et doit être défini au format "<description 1>|<extension 1>|<description 2>|<extension 2>...", par exemple, "Fichiers texte (*.txt)|*.txt|Tous les fichiers (*.*)|*.*". La première extension "Fichiers texte (*.txt)|*.txt" est sélectionnée comme type de fichier par défaut.
Si *filter*=NULL, le masque de sélection des fichiers dans la boîte de dialogue est "Tous les fichiers (*.*)|*.*|"

Si le paramètre *default_filename* est défini, un appel à FileSelectDialog() retourne 1, tandis que la valeur *default_filename* elle-même est copiée dans le tableau *filenames[]* pendant les tests non visuels.

La fonction ne peut pas être utilisée dans les indicateurs personnalisés car l'appel de FileSelectDialog() suspend le thread d'exécution pendant tout le temps en attendant la réponse de l'utilisateur. Étant donné que tous les indicateurs pour chaque symbole sont exécutés dans un seul thread, une telle suspension rend le fonctionnement de tous les graphiques sur toutes les périodes pour ce symbole impossible.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- récupère les noms des fichiers texte à télécharger depuis le dossier commun des
   string filenames[];
   if(FileSelectDialog("Sélectionner les fichiers à télécharger", NULL,
                       "Fichiers texte (*.txt)|*.txt|Tous les fichiers (*.*)|*.*",
                       FSD_ALLOW_MULTISELECT|FSD_COMMON_FOLDER, filenames, "data.txt")
     {
      //--- affiche le nom de chaque fichier sélectionné
      int total=ArraySize(filenames);
      for(int i=0; i<total; i++)
         Print(i, ": ", filenames[i]);
     }
   else
     {
     Print("Aucun fichier sélectionné");
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

Voir aussi

FileOpen, FileIsExist, FileDelete, FileMove, FolderCreate, FolderDelete, FolderClean, Flags d'ouverture des fichiers

© 2000-2024, MetaQuotes Ltd.

FileFindFirst

Commence le balayage des fichiers et des sous-répertoires dans le directoire correspondant conformément au filtre indiqué.

```
long  FileFindFirst(
   const string   file_filter,          // chaîne - le filtre de la recherche
   string&        returned_filename,    // le nom du fichier trouvé ou du sous-répert
   int            common_flag=0         // définit le domaine de recherches
   );
```

Paramètres

file_filter

[in] Le filtre de la recherche. On peut indiquer le sous-répertoire dans le filtre (ou la séquence des sous-répertoires inclus) concernant le répertoire \Files, où il est nécessaire de passer l'excédent des fichiers.

returned_filename

[out] Le paramètre rendu, où en cas de la réussite se place le nom du premier fichier trouvé ou du sous-répertoire. Only the file name is returned (including the extension), the directories and subdirectories are not included no matter if they are specified or not in the search filter.

common_flag

[in] Le drapeau, déterminant l'endroit du fichier. Si common_flag=FILE_COMMON, le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le fichier se trouve dans le dossier local.

La valeur rendue

Rend le handle de l'objet de la recherche, lequel il est nécessaire d'utiliser pour un balayage ultérieur des fichiers et des sous-répertoires par la fonction FileFindNext(), ou INVALID_HANDLE dans le cas où il n'y a pas d'aucun fichier et le sous-répertoire, correspondant au filtre (dans le cas particulier - le répertoire est vide). Après la fin de la recherche il est nécessaire de fermer le handle à l'aide de la fonction FileFindClose().

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//--- display the window of input parameters when launching the script
#property script_show_inputs
//--- filter
input string InpFilter="Dir1\\*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   string int_dir="";
   int    i=1,pos=0,last_pos=-1;
//--- search for the last backslash
   while(!IsStopped())
     {
      pos=StringFind(InpFilter,"\\",pos+1);
      if(pos>=0)
         last_pos=pos;
      else
         break;
     }
//--- the filter contains the folder name
   if(last_pos>=0)
      int_dir=StringSubstr(InpFilter,0,last_pos+1);
//--- get the search handle in the root of the local folder
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- check if the FileFindFirst() is executed successfully
   if(search_handle!=INVALID_HANDLE)
     {
      //--- in a cycle, check if the passed strings are the names of files or directo
      do
        {
         ResetLastError();
         //--- if it's a file, the function returns true, and if it's a directory, it
         FileIsExist(int_dir+file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "D
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- close the search handle
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

Voir aussi

FileFindNext, FileFindClose

© 2000-2024, MetaQuotes Ltd.

FileFindNext

Continue la recherche commencée par la fonction FileFindFirst().

```
bool  FileFindNext(
   long      search_handle,         // handle de la recherche
   string&   returned_filename      // le nom du fichier trouvé ou du sous-répertoire
   );
```

Paramètres

search_handle

[in] Le handle de la recherche, reçu par la fonction FileFindFirst().

returned_filename

[out] Le nom du fichier suivant trouvé ou du répertoire. Only the file name is returned (including the extension), the directories and subdirectories are not included no matter if they are specified or not in the search filter.

La valeur rendue

Rend true en cas du succès, autrement rend false.

Exemple:

```
//--- affichons la fenêtre des paramètres d'entrée pendant le lancement du script
#property script_show_inputs
//--- le filtre
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- la réception du handle de la recherche dans la racine du dossier local
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- vérifions si la fonction FileFindFirst() a travaillé avec succès
   if(search_handle!=INVALID_HANDLE)
     {
      //--- dans le cycle vérifions si les chaînes transmises sont les noms des fichie
      do
        {
         ResetLastError();
         //--- si c'est le fichier, la fonction rendra true, et si c'est le répertoire
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "D
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- fermons le handle de la recherche
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

© 2000-2024, MetaQuotes Ltd.

Voir aussi

FileFindFirst, FileFindClose

© 2000-2024, MetaQuotes Ltd.

FileFindClose

Ferme le handle de la recherche.

```
void  FileFindClose(
   long  search_handle      // handle de la recherche
   );
```

Paramètres

search_handle

[in] Le handle de la recherche, reçu par la fonction FileFindFirst().

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Il est nécessaire d'appeler la fonction pour désallouer les ressources systémiques.

Exemple:

```
//--- affichons la fenêtre des paramètres d'entrée pendant le lancement du script
#property script_show_inputs
//--- le filtre
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- la réception du handle de la recherche dans la racine du dossier local
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- vérifions si la fonction FileFindFirst() a travaillé avec succès
   if(search_handle!=INVALID_HANDLE)
     {
      //--- dans le cycle vérifions si les chaînes transmises sont les noms des fichie
      do
        {
         ResetLastError();
         //--- si c'est le fichier, la fonction rendra true, et si c'est le répertoire
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==5018 ? "Directory" : "File"
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- fermons le handle de la recherche
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

Voir aussi

FileFindFirst, FileFindNext

© 2000-2024, MetaQuotes Ltd.

FileIsExist

Vérifie l'existence d'un fichier.

```
bool  FileIsExist(
   const string  file_name,        // nom du fichier
   int           common_flag=0     // domaine de recherche
   );
```

Paramètres

file_name

[in] Le nom du fichier vérifié.

common_flag=0

[in] Le drapeau, déterminant l'endroit du fichier. Si common_flag=FILE_COMMON, le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le fichier se trouve dans le dossier local.

La valeur rendue

Rend true, si le fichier indiqué existe.

Note

Le fichier vérifié peut être le sous-répertoire. Dans ce cas la fonction FileIsExist() rendra false, et l'erreur 5018 - "Ce n'est pas un fichier, c'est un répertoire" (regardez l'exemple pour la fonction FileFindFirst) sera inscrite dans une variable _LastError.

On contrôle strictement le travail avec les fichiers dans la langue MQL5 pour des raisons de sécurité. Les fichiers, avec lesquels on mène des opérations de fichier par les moyens de la langue MQL5, ne peuvent pas se trouver en dehors du sandbox de fichier".

Si common_flag=FILE_COMMON, la fonction cherche le fichier indiqué dans le dossier commun de tous les terminaux de client \Terminal\Common\Files, dans le cas contraire la fonction cherche le fichier dans le dossier local (MQL5\Files ou MQL5\Tester\Files en cas du test).

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//---la date pour les anciens fichiers
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // la variable pour stocker les noms des fichiers
   string   filter="*.txt"; // le filtre pour rechercher les fichiers
   datetime create_date;    // la date de la création du fichier
   string   files[];        // la liste des noms des fichiers
   int      def_size=25;    // la taille du tableau par défaut
   int      size=0;         // le nombre de fichiers
//--- allouons la mémoire pour un tableau
   ArrayResize(files,def_size);
//--- la réception du handle de la recherche dans la racine du dossier local
   long search_handle=FileFindFirst(filter,file_name);
//--- vérifions, si la fonction FileFindFirst() a travaillé avec succès
   if(search_handle!=INVALID_HANDLE)
     {
      //--- trions les fichiers dans le cycle
      do
        {
         files[size]=file_name;
         //--- augmentons la taille du tableau
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- oblitérons la valeur de l'erreur
         ResetLastError();
         //---recevrons la date de la création du fichier
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- vérifions si le fichier est ancien
         if(create_date<InpFilesDate)
           {
            PrintFormat("Le fichier %s est supprimé!",file_name);
            //--- supprimons un ancien fichier
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- fermons le handle de la recherche
      FileFindClose(search_handle);
     }
   else
     {
      Print("Files not found!");
      return;
     }
//--- vérifions quels fichiers sont restés
   PrintFormat("Les résultats:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("Le fichier %s existe!",files[i]);
      else
```

© 2000-2024, MetaQuotes Ltd.

```
            PrintFormat("Le fichier %s est supprimé!",files[i]);
        }
    }
```

Voir aussi

FileFindFirst

© 2000-2024, MetaQuotes Ltd.

FileOpen

La fonction ouvre le fichier avec le nom et les flags spécifiés.

```
int  FileOpen(
   string  file_name,          // Nom du fichier
   int     open_flags,         // Combinaison de flags
   short   delimiter='\t',     // Délimiteur
   uint    codepage=CP_ACP     // Page de code
   );
```

Paramètres

file_name

[in] Le nom du fichier, peut contenir des sous-dossiers. Si le fichier est ouvert en écriture, ces sous-dossiers seront créés s'ils n'existent pas.

open_flags

[in] combinaison de flags déterminant le mode d'utilisation du fichier. Les flags sont définis comme suit :
FILE_READ le fichier est ouvert en lecture
FILE_WRITE le fichier est ouvert en écriture
FILE_BIN lecture-écriture en mode binaire (aucune conversion depuis et vers une chaîne de caractères)
FILE_CSV fichier de type csv (touts les élements sont convertis en chaînes de caractères de type unicode ou ansi, et sont séparés par un délimiteur)
FILE_TXT un fichier texte simple (le même que csv, mais le délimiteur n'est pas pris en compte)
FILE_ANSI lignes de type ANSI (symboles sur seul octet)
FILE_UNICODE lignes de type UNICODE (caractères sur 2 octets)
FILE_SHARE_READ lecture partagée sur plusieurs programmes
FILE_SHARE_WRITE écriture partagée sur plusieurs programmes
FILE_COMMON localisation du fichier dans un répertoire partagé pour tous les terminaux clients \Terminal\Common\Files

delimiter='\t'

[in] valeur à utiliser comme séparateur pour les fichiers txt et csv. Si le délimiteur n'est pas spécifié pour le fichier csv, la valeur par défaut est une tabulation. Si le délimiteur n'est pas spécifié pour le fichier txt, aucun séparateur n'est utilisé. Si le séparateur est mis à 0, aucun séparateur n'est utilisé.

codepage=CP_ACP

[in] La valeur de la page de code. Pour les pages de code les plus utilisées, il existe des constantes correspondantes.

Valeur de Retour

Si un fichier a été ouvert avec succès, la fonction retourne un handle sur le fichier, qui est ensuite utilisé pour accéder aux données du fichier. En cas d'échec, retourne INVALID_HANDLE.

Note

© 2000-2024, MetaQuotes Ltd.

Pour des raisons de sécurité, travailler avec les fichiers est contrôlé de façon stricte dans le langage MQL5. Les fichiers utilisés par des opérations MQL5 ne peuvent pas être en dehors du sandbox des fichiers.

Assurez-vous de spécifier le flag FILE_ANSI si le fichier doit être lu avec un encodage spécifique (le paramètre codepage avec une page de code est spécifié). Si le flag FILE_ANSI n'est pas utilisé, le fichier texte est lu en Unicode sans conversion.

Le fichier est ouvert depuis le dossier du terminal client dans le sous-dossier MQL5\files (ou testing_agent_directory\MQL5\files en cas de test). Si FILE_COMMON est spécifié dans les flags, le fichier est ouvert dans un répertoire partagé entre tous les terminaux clients MetaTrader 5.

Les "Pipes nommés" peuvent être ouverts suivant les règles suivantes :

- Le nom d'un pipe est une chaîne de caractères au format : "\\servername\pipe\pipename", où servername - le nom du serveur sur le réseau, et pipename est le nom d'un pipe. Si les pipes sont utilisés sur le même ordinateur, le nom du serveur peut être omis, mais un point doit être inséré à sa place : "\\.\pipe\pipename". Un client qui essaye de connecter un pipe doit connaître son nom.
- FileFlush() et FileSeek() doivent être appelés au début du fichier entre les opérations séquentielles de lecture depuis le pipe et l'écriture dans celui-ci.

Le symbole spécial '\' est utilisé dans les chaînes de caractères affichées. Donc, '\' doit être doublé pour écrire un nom dans une application MQL5. Cela signifie que l'exemple ci-dessus doit ressembler dans le code à : "\\\\servername\\pipe\\pipename".

Plus d'informations sur l'utilisation des pipes nommés sont disponibles dans l'article "Communicating With MetaTrader 5 Using Named Pipes Without Using DLLs".

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Méthode incorrecte d'ouverture de fichier
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   string filename=terminal_data_path+"\\MQL5\\Files\\"+"fractals.csv";
   int filehandle=FileOpen(filename,FILE_WRITE|FILE_CSV);
   if(filehandle<0)
     {
      Print("Echec de l'ouverture du fichier avec un chemin absolu ");
      Print("Code d'erreur ",GetLastError());
     }

//--- Façon correcte de travailler dans le 'sandbox des fichiers'
   ResetLastError();
   filehandle=FileOpen("fractals.csv",FILE_WRITE|FILE_CSV);
   if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
```

© 2000-2024, MetaQuotes Ltd.

```
      Print("FileOpen OK");
     }
   else Print("Echec de l'opération FileOpen, erreur ",GetLastError());
//--- un autre exemple de création d'un répertoire dans MQL5\Files\
   string subfolder="Research";
   filehandle=FileOpen(subfolder+"\\fractals.txt",FILE_WRITE|FILE_CSV);
      if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("Le fichier doit être créé dans le dossier "+terminal_data_path+"\\"+subfo
     }
   else Print("Echec d'ouverture du fichier, erreur ",GetLastError());
  }
```

Voir aussi

Utilisation d'une page de code, FileFindFirst, FolderCreate, Flags d'ouverture d'un fichier

© 2000-2024, MetaQuotes Ltd.

FileClose

Ferme le fichier auparavant ouvert par la fonction FileOpen().

```
void  FileClose(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendue par la fonction FileOpen().

La valeur rendue

Il n'y a pas de valeur rendue.

Exemple:

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="file.txt";     // le nom du fichier
input string InpDirectoryName="Data";    // le nom du répertoire
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprimons le chemin vers le dossier dans lequel on va travailler
   PrintFormat("Travaillons dans le dossier %s\\Files\\",TerminalInfoString(TERMINAL_I
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrirons le fichier pour la lecture (si le fichier n'existe pas, une erreur se
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //---imprimons le contenu du fichier
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
      PrintFormat("L'erreur, le code = %d",GetLastError());
  }
```

© 2000-2024, MetaQuotes Ltd.

FileCopy

Copie le fichier original d'un répertoire local ou commun à un autre fichier.

```
bool  FileCopy(
   const string  src_file_name,      // nom du fichier-source
   int           common_flag,        // lieu de l'action
   const string  dst_file_name,      // nom du fichier de la destination
   int           mode_flags          // mode de l'accès
   );
```

Paramètres

src_file_name

[in] Le nom du fichier pour le copiage.

common_flag

[in] Le drapeau, déterminant l'endroit du fichier. Si common_flag=FILE_COMMON, le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le fichier se trouve dans le dossier local (par exemple, *common_flag=0*).

dst_file_name

[in] Le nom de fichier de résultat.

mode_flags

[in] Les drapeaux de l'accès. Le paramètre peut contenir seulement 2 drapeaux: FILE_REWRITE et/ou FILE_COMMON - les autres drapeaux sont ignorés. Si le fichier existe déjà et le drapeau FILE_REWRITE n'a pas été spécifié, donc le fichier ne sera pas réécrit et la fonction rendra false.

La valeur rendue

En cas de l'échec la fonction rend false.

Note

Si un nouveau fichier existait, le copiage sera produit en fonction de la présence du drapeau FILE_REWRITE dans la valeur du paramètre mode_flags.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//---affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpSrc="source.txt";       // la source
input string InpDst="destination.txt";  // la copie
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- affichons le contenu de la source (il doit exister)
   if(!FileDisplay(InpSrc))
      return;
//--- vérifions, s'il y a déjà un fichier de la copie (ne doit pas être créé)
   if(!FileDisplay(InpDst))
     {
      //--- le fichier de la copie n'existe pas, le copiage sans drapeau FILE_REWRITE
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("File is copied!");
      else
         Print("File is not copied!");
     }
   else
     {
      //--- le fichier de la copie existe déjà, essaierons copier sans drapeau FILE_RE
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("File is copied!");
      else
         Print("File is not copied!");
      //---le contenu du fichier InpDst sera le même
      FileDisplay(InpDst);
      //--- copions encore une fois avec le drapeau FILE_REWRITE (le copiage juste à l
      if(FileCopy(InpSrc,0,InpDst,FILE_REWRITE))
         Print("File is copied!");
      else
         Print("File is not copied!");
     }
//--- ont reçu la copie du fichier InpSrc
   FileDisplay(InpDst);
  }
//+------------------------------------------------------------------+
//| la lecture du contenu du fichier                                 |
//+------------------------------------------------------------------+
bool FileDisplay(const string file_name)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrons le fichier
   int file_handle=FileOpen(file_name,FILE_READ|FILE_TXT|InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- affichons le contenu d'un fichier dans une boucle
      Print("+----------------------+");
      PrintFormat("File name = %s",file_name);
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      Print("+----------------------+");
      //--- fermons le fichier
      FileClose(file_handle);
      return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- on n'a pas réussi à ouvrir le fichier
   PrintFormat("%s is not opened, error = %d",file_name,GetLastError());
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

FileDelete

Supprime le fichier indiqué dans le dossier local du terminal de client.

```
bool  FileDelete(
   const string  file_name,      // nom du fichier supprimé
   int           common_flag=0   // endroit du fichier supprimé
   );
```

Paramètres

file_name

[in] Le nom du fichier.

common_flag=0

[in] Le drapeau, déterminant l'endroit du fichier. Si common_flag=FILE_COMMON, le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le fichier se trouve dans le dossier local.

La valeur rendue

En cas de l'échec la fonction rend false.

Note

On contrôle strictement le travail avec les fichiers dans la langue MQL5 pour des raisons de sécurité. Les fichiers, avec lesquels on mène des opérations de fichier par les moyens de la langue MQL5, ne peuvent pas se trouver en dehors du sandbox de fichier".

Supprime le fichier indiqué dans le dossier local du terminal de client (MQL5\Files ou MQL5\Tester\Files en cas du test). Si common_flag=FILE_COMMON est indiqué, la fonction supprime le fichier du dossier total de tous les terminaux de client.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//---la date pour les anciens fichiers
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // la variable pour stocker les noms des fichiers
   string   filter="*.txt"; // le filtre pour rechercher les fichiers
   datetime create_date;    // la date de la création du fichier
   string   files[];        // la liste des noms des fichiers
   int      def_size=25;    // la taille du tableau par défaut
   int      size=0;         // le nombre de fichiers
//--- allouons la mémoire pour un tableau
   ArrayResize(files,def_size);
//--- la réception du handle de la recherche dans la racine du dossier local
   long search_handle=FileFindFirst(filter,file_name);
//--- vérifions, si la fonction FileFindFirst() a travaillé avec succès
   if(search_handle!=INVALID_HANDLE)
     {
      //--- trions les fichiers dans le cycle
      do
        {
         files[size]=file_name;
         //--- augmentons la taille du tableau
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- oblitérons la valeur de l'erreur
         ResetLastError();
         //---recevrons la date de la création du fichier
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- vérifions si le fichier est ancien
         if(create_date<InpFilesDate)
           {
            PrintFormat("Le fichier %s est supprimé!",file_name);
            //--- supprimons un ancien fichier
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- fermons le handle de la recherche
      FileFindClose(search_handle);
     }
   else
     {
      Print("Files not found!");
      return;
     }
//--- vérifions quels fichiers sont restés
   PrintFormat("Les résultats:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("Le fichier %s existe!",files[i]);
      else
```

© 2000-2024, MetaQuotes Ltd.

```
        PrintFormat("Le fichier %s est supprimé!",files[i]);
    }
}
```

© 2000-2024, MetaQuotes Ltd.

FileMove

Déplace le fichier du dossier local ou total à un autre dossier.

```
bool  FileMove(
   const string  src_file_name,     // nom du fichier pour l'opération du déplacement
   int           common_flag,       // lieu de l'action
   const string  dst_file_name,     // nom du fichier de la destination
   int           mode_flags         // mode de l'accès
   );
```

Paramètres

src_file_name

[in] Le nom du fichier pour le déplacement/la renomination.

common_flag

[in] Le drapeau, déterminant l'endroit du fichier. Si common_flag=FILE_COMMON, le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le fichier se trouve dans le dossier local (*common_flag=0*).

dst_file_name

[in] Le nom de fichier de résultat.

mode_flags

[in] Les drapeaux de l'accès. Le paramètre peut contenir seulement 2 drapeaux: FILE_REWRITE et/ou FILE_COMMON - les autres drapeaux sont ignorés. Si le fichier existe déjà et le drapeau FILE_REWRITE, n'a pas été spécifié, donc le fichier ne sera pas réécrit et la fonction rendra false.

La valeur rendue

En cas de l'échec la fonction rend false.

Note

On contrôle strictement le travail avec les fichiers dans la langue MQL5 pour des raisons de sécurité. Les fichiers, avec lesquels on mène des opérations de fichier par les moyens de la langue MQL5, ne peuvent pas se trouver en dehors du sandbox de fichier".

Si un nouveau fichier existait, le copiage sera produit en fonction de la présence du drapeau FILE_REWRITE dans la valeur du paramètre mode_flags.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpSrcName="data.txt";
input string InpDstName="newdata.txt";
input string InpSrcDirectory="SomeFolder";
input string InpDstDirectory="OtherFolder";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string local=TerminalInfoString(TERMINAL_DATA_PATH);
   string common=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- recevrons les voies vers les fichiers
   string src_path;
   string dst_path;
   StringConcatenate(src_path,InpSrcDirectory,"//",InpSrcName);
   StringConcatenate(dst_path,InpDstDirectory,"//",InpDstName);
//--- vérifions s'il y a un fichier source (si elle n'existe pas - la sortie)
   if(FileIsExist(src_path))
      PrintFormat("%s file exists in the %s\\Files\\%s folder",InpSrcName,local,InpSrc
   else
     {
      PrintFormat("Error, %s source file not found",InpSrcName);
      return;
     }
//--- vérifions, s'il y a déjà un fichier du résultat
   if(FileIsExist(dst_path,FILE_COMMON))
     {
      PrintFormat("%s file exists in the %s\\Files\\%s folder",InpDstName,common,InpDs
      //--- le fichier existe, il faut passer le déplacement avec le drapeau FILE_REWR
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON|FILE_REWRITE))
         PrintFormat("%s file moved",InpSrcName);
      else
         PrintFormat("Error! Code = %d",GetLastError());
     }
   else
     {
      PrintFormat("%s file does not exist in the %s\\Files\\%s folder",InpDstName,comm
      //--- le fichier n'existe pas, il faut passer le déplacement sans drapeau FILE_R
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON))
         PrintFormat("%s file moved",InpSrcName);
      else
         PrintFormat("Error! Code = %d",GetLastError());
     }
//--- maintenant le fichier est déplacé, vérifions ça
   if(FileIsExist(dst_path,FILE_COMMON) && !FileIsExist(src_path,0))
      Print("Success!");
   else
      Print("Error!");
  }
```

Voir aussi

FileIsExist

© 2000-2024, MetaQuotes Ltd.

FileFlush

Écrit à un disque toutes les données restant dans le tampon du fichier d'entrée-de sortie.

```
void  FileFlush(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

Il n'y a pas de valeur rendue.

Note

A l'exécution de l'opération de l'inscription au fichier, les données peuvent s'y trouver physiquement seulement dans un certain temps. Pour que les données se sauvegardent tout de suite dans un fichier, il faut utiliser la fonction FileFlush(). Si ne pas utiliser la fonction, la partie des données qui ne se sont pas encore sur le disque s'inscrit forcément là-bas seulement à la clôture du fichier par la fonction FileClose().

Il est nécessaire d'utiliser la fonction, quand les données inscrites ont la valeur définie. Il faut prendre en considération que l'appel fréquent de la fonction peut affecter la vitesse du travail du programme.

Il est nécessaire d'appeler la fonction FileFlush() entre les opérations de la lecture du fichier et l'enregistrement au fichier.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//---le nom du fichier pour l'inscription
input string InpFileName="example.csv"; // le nom du fichier
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrons le fichier
   int file_handle=FileOpen(InpFileName,FILE_READ|FILE_WRITE|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- inscrirons les données au fichier
      for(int i=0;i<1000;i++)
        {
         //--- appelons la fonction de l'inscription
         FileWrite(file_handle,TimeCurrent(),SymbolInfoDouble(Symbol(),SYMBOL_BID),Sym
         //--- enregistrons les données sur le disque sur chaque 128-ième itération
         if((i & 127)==127)
           {
            //--- maintenant les données se trouvent dans le fichier, et à l'erreur ca
            FileFlush(file_handle);
            PrintFormat("i = %d, OK",i);
           }
         //--- le retard à 0.01 seconde
         Sleep(10);
        }
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
      PrintFormat("L'erreur, le code = %d",GetLastError());
  }
```

Voir aussi

FileClose

© 2000-2024, MetaQuotes Ltd.

FileGetInteger

Reçoit une propriété entière du fichier. Il y a 2 variantes de la fonction.

1. La réception des propriétés selon le handle du fichier.

```
long  FileGetInteger(
   int                          file_handle,    // le handle du fichier
   ENUM_FILE_PROPERTY_INTEGER   property_id     // l'identificateur de la propriété
   );
```

2. La réception des propriétés selon le nom du fichier.

```
long  FileGetInteger(
   const string                 file_name,            // le nom du fichier
   ENUM_FILE_PROPERTY_INTEGER   property_id,          // l'identificateur de la proprié
   bool                         common_folder=false   // le fichier est affiché dans ur
   );                                                 // ou dans le dossier commun de t
```

Paramètres

file_handle

[in] Le descripteur de fichier, rendu par la fonction FileOpen().

file_name

[in] Le nom du fichier.

property_id

[in] L'identificateur de la propriété du fichier. La valeur peut être l'une des valeurs de l'énumérationENUM_FILE_PROPERTY_INTEGER. Si on utilise une deuxième variante de la fonction, vous pouvez recevoir la valeur seulement des propriétés suivantes: FILE_EXISTS, FILE_CREATE_DATE, FILE_MODIFY_DATE, FILE_ACCESS_DATE et FILE_SIZE.

common_folder=false

[in] Indique à l'emplacement du fichier. Si le paramètre est égal false, on examine le répertoire des données du terminal, sinon il est supposé que le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files (FILE_COMMON).

La valeur rendue

La valeur de la propriété. En cas de l'erreur rend -1, pour recevoir un code de l'erreur il faut appeler la fonction GetLastError().

Si on indique le répertoire à la réception des propriétés selon le nom, la fonction mettra l'erreur 5018 (ERR_MQL_FILE_IS_DIRECTORY) dans tous les cas, en cela la valeur rendue sera correcte.

Note

La fonction change toujours le code de l'erreur. En cas de succès, le code de l'erreur est remis à zéro.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="data.csv";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string path=InpDirectoryName+"//"+InpFileName;
   long   l=0;
//--- ouvrons le fichier
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_CSV);
   if(handle!=INVALID_HANDLE)
     {
      //--- imprimons toute l'information sur le fichier
      Print(InpFileName," file info:");
      FileInfo(handle,FILE_EXISTS,l,"bool");
      FileInfo(handle,FILE_CREATE_DATE,l,"date");
      FileInfo(handle,FILE_MODIFY_DATE,l,"date");
      FileInfo(handle,FILE_ACCESS_DATE,l,"date");
      FileInfo(handle,FILE_SIZE,l,"other");
      FileInfo(handle,FILE_POSITION,l,"other");
      FileInfo(handle,FILE_END,l,"bool");
      FileInfo(handle,FILE_IS_COMMON,l,"bool");
      FileInfo(handle,FILE_IS_TEXT,l,"bool");
      FileInfo(handle,FILE_IS_BINARY,l,"bool");
      FileInfo(handle,FILE_IS_CSV,l,"bool");
      FileInfo(handle,FILE_IS_ANSI,l,"bool");
      FileInfo(handle,FILE_IS_READABLE,l,"bool");
      FileInfo(handle,FILE_IS_WRITABLE,l,"bool");
      //--- fermons le fichier
      FileClose(handle);
     }
   else
      PrintFormat("%s file is not opened, ErrorCode = %d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------+
//| L'affichage de la valeur des propriétés du fichier               |
//+------------------------------------------------------------------+
void FileInfo(const int handle,const ENUM_FILE_PROPERTY_INTEGER id,
              long l,const string type)
  {
//--- recevrons la valeur de la propriété
   ResetLastError();
   if((l=FileGetInteger(handle,id))!=-1)
     {
      //--- La valeur est reçue, l'affichons au format juste
      if(!StringCompare(type,"bool"))
         Print(EnumToString(id)," = ",l ? "true" : "false");
      if(!StringCompare(type,"date"))
         Print(EnumToString(id)," = ",(datetime)l);
      if(!StringCompare(type,"other"))
         Print(EnumToString(id)," = ",l);
     }
   else
      Print("Error, Code = ",GetLastError());
  }
```

© 2000-2024, MetaQuotes Ltd.

Voir aussi

Les opérations de fichier, Les propriétés des fichiers

© 2000-2024, MetaQuotes Ltd.

FileIsEnding

Définit la fin du fichier en train de la lecture.

```
bool  FileIsEnding(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La fonction rend true dans le cas, si en train de la lecture ou du déplacement de l'indication de fichier atteindront la fin du fichier.

Note

Pour la définition de la fin du fichier, la fonction essaie de lire la chaîne suivante du fichier. Si elle n'existe pas, la fonction rend true, autrement rend false.

Exemple:

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="file.txt";    // le nom du fichier
input string InpDirectoryName="Data";   // le nom du répertoire
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprimons le chemin vers le dossier dans lequel on va travailler
   PrintFormat("Travaillons dans le dossier %s\\Files\\",TerminalInfoString(TERMINAL_I
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrirons le fichier pour la lecture (si le fichier n'existe pas, une erreur se
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //---imprimons le contenu du fichier
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
      PrintFormat("L'erreur, le code = %d",GetLastError());
  }
```

© 2000-2024, MetaQuotes Ltd.

FileIsLineEnding

Définit la fin de la ligne dans le fichier du texte en train de la lecture.

```
bool  FileIsLineEnding(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La fonction rend true dans le cas, si en train de la lecture des fichiers txt ou csv atteint la fin de la ligne (les caractères CR-LF).

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteString)

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileIsLineEnding.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- les paramètres pour la lecture des données
input string InpFileName="RSI.csv";   // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les tampons d'indicateur
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- les variables du surachat
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- les variables de la survente
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- les variables des tailles des tableaux par défaut
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- allouons la mémoire pour les tableaux
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier : %s\\Files\\",TerminalInfoString(TERMINAL_
      double value;
      //--- lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //---lisons la première signification dans la chaîne
         value=FileReadNumber(file_handle);
         //--- lisons aux tableaux différents en fonction du résultat de la fonction
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
```

```
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- le rattachement des tableaux
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
//---- l'établissement des valeurs de l'indicateur, qui ne seront pas vus sur le graph
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| La lecture des données de la chaîne du fichier
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- lisons jusqu'à ce que nous arrivons à la fin d'une chaîne ou d'un fichier
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- rapprocherons le chariot, ayant lu le nombre
      if(flag)
         FileReadNumber(file_handle);
      //--- retenons la date courante
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- augmentons en cas de nécessité la taille du tableau
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- on a passé la première itération
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2024, MetaQuotes Ltd.

```
{
```

© 2000-2024, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- si les dates correspondent, la barre se trouve dans la zone de surac
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - la couleur rouge
               color_buff[i]=0;
               //--- augmentons le compteur
               ovb_ind=j+1;
               break;
              }
           }
      //--- la vérification s'il y a encore des données
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- si les dates correspondent, la barre se trouve dans la zone de la su
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - la couleur bleue
               color_buff[i]=1;
               //---augmentons le compteur
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Le gestionnaire de l'événement ChartEvent                        |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
```

© 2000-2024, MetaQuotes Ltd.

```
                    )
  {
//---changeons l'épaisseur de l'indicateur en fonction de l'échelle
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Voir aussi

FileWriteString

© 2000-2024, MetaQuotes Ltd.

FileReadArray

Lit les tableaux de tous les types, sauf les tableaux de chaîne (peut être un tableau des structures qui ne contient pas les chaînes et les tableaux dynamiques), du fichier binaire de la position courante de l'indicateur de fichier.

```
uint  FileReadArray(
   int     file_handle,                // handle du fichier
   void&   array[],                    // tableau pour l'enregistrement
   int     start=0,                    // de quelle position du tableau écrire
   int     count=WHOLE_ARRAY           // combien de lire
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

array[]

[out] Le tableau où les données seront chargées.

start=0

[in] La position de départ pour l'enregistrement au tableau.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour la lecture.

La valeur rendue

Le nombre d'éléments lus. Par défaut, lit tout le tableau (count=WHOLE_ARRAY).

Note

Le tableau de chaîne peut être lu seulement du fichier du type TXT. En cas de nécessité la fonction tente d'augmenter la grandeur du tableau.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteArray)

© 2000-2024, MetaQuotes Ltd.

```
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| La structure pour la sauvegarde des données sur les prix
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // la date
   double            bid;  // le prix bid
   double            ask;  // le prix ask
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- un tableau de structure
   prices arr[];
//--- la voie vers le fichier
   string path=InpDirectoryName+"//"+InpFileName;
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(path,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- lisons toutes les données du fichier au tableau
      FileReadArray(file_handle,arr);
      //--- recevrons la taille du tableau
      int size=ArraySize(arr);
      //--- imprimons les données du tableau
      for(int i=0;i<size;i++)
         Print("Date = ",arr[i].date," Bid = ",arr[i].bid," Ask = ",arr[i].ask);
      Print("Total data = ",size);
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
      Print("File open failed, error ",GetLastError());
  }
```

Voir aussi

Les variables, FileWriteArray

© 2000-2024, MetaQuotes Ltd.

FileReadBool

Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type bool.

```
bool  FileReadBool(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La ligne lue peut avoir les valeurs "true", "false" ou la représentation symbolique des nombres entières "0" ou "1". La valeur non zéro se transforme vers logique true. La fonction rend la valeur reçue transformée.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWrite)

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              Demo_FileReadBool.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- les paramètres pour la lecture des données
input string InpFileName="MACD.csv";  // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les variables globales
int      ind=0;       // l'index
double   upbuff[];    // le tampon d'indicateur des flèches en haut
double   downbuff[];  // le tampon d'indicateur des flèches en bas
bool     sign_buff[]; // le tableau des signaux (true - l'achat, false - la vente)
datetime time_buff[]; // le tableau du temps de l'arrivée des signaux
int      size=0;      // la taille des tableau des signaux
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      //--- lisons d'abord le nombre de signaux
      size=(int)FileReadNumber(file_handle);
      //--- allouons la mémoire pour les tableaux
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- lisons des données du fichier
      for(int i=0;i<size;i++)
        {
         //--- le temps du signal
         time_buff[i]=FileReadDatetime(file_handle);
         //--- la valeur du signal
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
```

```
      {
       PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
       return(INIT_FAILED);
      }
//--- le rattachement des tableaux
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- spécifions le code du symbole pour la rendu dans PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- l'établissement des valeurs de l'indicateur qui ne seront pas visibles sur le g
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               //--- dessinons la flèche en fonction du signal
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Le type bool, FileWrite

© 2000-2024, MetaQuotes Ltd.

FileReadDatetime

Lit du fichier du type CSV la chaîne d'un des formats: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" ou "HH:MI:SS" - et la transforme en valeur du type datetime.

```
datetime  FileReadDatetime(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type datetime.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWrite)

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileReadDateTime.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- les paramètres pour la lecture des données
input string InpFileName="MACD.csv";  // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les variables globales
int      ind=0;       // l'index
double   upbuff[];    // le tampon d'indicateur des flèches en haut
double   downbuff[];  // le tampon d'indicateur des flèches en bas
bool     sign_buff[]; // le tableau des signaux (true - l'achat, false - la vente)
datetime time_buff[]; // le tableau du temps de l'arrivée des signaux
int      size=0;      // la taille des tableau des signaux
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      //--- lisons d'abord le nombre de signaux
      size=(int)FileReadNumber(file_handle);
      //--- allouons la mémoire pour les tableaux
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- lisons des données du fichier
      for(int i=0;i<size;i++)
        {
         //--- le temps du signal
         time_buff[i]=FileReadDatetime(file_handle);
         //--- la valeur du signal
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
```

```
      {
       PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %c
       return(INIT_FAILED);
      }
//--- le rattachement des tableaux
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- spécifions le code du symbole pour la rendu dans PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- l'établissement des valeurs de l'indicateur qui ne seront pas visibles sur le g
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               //--- dessinons la flèche en fonction du signal
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Le type datetime, StringToTime, TimeToString, FileWrite

© 2000-2024, MetaQuotes Ltd.

FileReadDouble

Lit le nombre de l'exactitude double avec une virgule flottante (double) du fichier binaire de la position courante de l'indicateur de fichier.

```
double  FileReadDouble(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type double.

Note

Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteDouble)

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- les paramètres pour la lecture des données
input string InpFileName="MA.csv";    // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les variables globales
int      ind=0;
int      size=0;
double   ma_buff[];
datetime time_buff[];
//--- indicator buffer
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- lisons d'abord combien de données sont dans le fichier
      size=(int)FileReadDouble(file_handle);
      //--- allouons la mémoire pour les tableaux
      ArrayResize(ma_buff,size);
      ArrayResize(time_buff,size);
      //--- lisons les données du fichier
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadDouble(file_handle);
         ma_buff[i]=FileReadDouble(file_handle);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- Le rattachement du tableau au tampon d'indicateur avec l'indice 0
```

```
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- l'établissement des valeurs de l'indicateur qui ne seront pas visibles sur le g
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      buff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               buff[i]=ma_buff[j];
               //---augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Les types réels (double, float), StringToDouble, DoubleToString, FileWriteDouble

© 2000-2024, MetaQuotes Ltd.

FileReadFloat

Lit le nombre de l'exactitude simple avec la virgule flottante (float) de la position courante du fichier binaire.

```
float  FileReadFloat(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type float.

Note

Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteFloat)

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "CloseLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- les paramètres pour la lecture des données
input string InpFileName="Close.bin"; // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les variables globales
int      ind=0;
int      size=0;
double   close_buff[];
datetime time_buff[];
//--- indicator buffers
double   buff[];
double   color_buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- allouons la mémoire pour les tableaux
   ArrayResize(close_buff,def_size);
   ArrayResize(time_buff,def_size);
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat(Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //--- lisons la signification du temps et du prix
         time_buff[size]=(datetime)FileReadDouble(file_handle);
         close_buff[size]=(double)FileReadFloat(file_handle);
         size++;
         //--- augmenons les tailles des tableaux s'ils sont remplis
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(close_buff,def_size);
            ArrayResize(time_buff,def_size);
           }
        }
      //--- fermons le fichier
      FileClose(file_handle);
```

```
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %c
      return(INIT_FAILED);
     }
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,buff,INDICATOR_DATA);
   SetIndexBuffer(1,color_buff,INDICATOR_COLOR_INDEX);
//---- l'établissement des valeurs de l'indicateur, qui ne seront pas vus sur le graph
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      buff[i]=0;
      color_buff[i]=0; // la couleur rouge par défaut
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               //--- recevrons le prix
               buff[i]=close_buff[j];
               //--- si le prix actuel est plus que le précédente, la couleur est bleu
               if(buff[i-1]>buff[i])
                  color_buff[i]=1;
               //--- augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Les types réels (double, float), FileReadDouble, FileWriteFloat

© 2000-2024, MetaQuotes Ltd.

FileReadInteger

Lit du fichier binaire la valeur du int, short ou char en fonction de la longueur indiquée dans les bytes. La lecture est produite de la position courante de l'indicateur de fichier.

```
int  FileReadInteger(
   int  file_handle,        // handle du fichier
   int  size=INT_VALUE      // taille d'un nombre entier en bytes
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

size=INT_VALUE

[in] Le nombre de bytes (jusqu'à 4 inclus) qu'il faut lire. On prévoit les constantes correspondantes: CHAR_VALUE=1, SHORT_VALUE=2 et INT_VALUE=4, ainsi la fonction peut lire la valeur entière comme char, short ou int.

La valeur rendue

La valeur du type int. Il est nécessaire évidemment d'amener le résultat de cette fonction au type cible, c'est-à-dire à ce type de données, qu'il faut lire. Puisque revient la valeur comme int, on peut le transformer en n'importe quelle valeur entière. Le pointeur de fichier est déplacé par le nombre de bytes lus.

Note

A la lecture de moins 4 bytes, le résultat acquis sera toujours positif. Si un ou deux bytes sont lus, on peut exactement définir le signe du nombre par voie de la réduction évidente en conséquence vers le type char (1 byte) ou le type short (2 bytes). La réception du signe pour le nombre de trois bytes est non triviale, puisqu'il n'y a pas de type de basecorrespondant.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteInteger)

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Trends"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres pour la lecture des données
input string InpFileName="Trend.bin"; // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les variables globales
int      ind=0;
int      size=0;
datetime time_buff[];
//--- indicator buffers
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- allouons la mémoire pour le tableau
   ArrayResize(time_buff,def_size);
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- les variables auxiliaires
      int    arr_size;
      uchar  arr[];
      //--- lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //--- apprenons combien de symboles sont utilisés pour l'inscription du temps
         arr_size=FileReadInteger(file_handle,INT_VALUE);
         ArrayResize(arr,arr_size);
         for(int i=0;i<arr_size;i++)
            arr[i]=(char)FileReadInteger(file_handle,CHAR_VALUE);
         //--- retiendrons la valeur du temps
         time_buff[size]=StringToTime(CharArrayToString(arr));
         size++;
         //--- augmentons les tailles des tableaux s'ils sont remplis
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(time_buff,def_size);
           }
```

```
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- Le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- l'établissement des valeurs de l'indicateur qui ne seront pas visibles sur le g
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(close,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      buff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //---si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               //--- recevrons le prix
               buff[i]=close[i];
               //--- augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

IntegerToString, StringToInteger, Les types entiers, FileWriteInteger

© 2000-2024, MetaQuotes Ltd.

FileReadLong

Lit le nombre entier du type long (8 bytes) de la position courante du fichier binaire.

```
long  FileReadLong(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type long.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteLong)

© 2000-2024, MetaQuotes Ltd.

```
//+--//+------------------------------------------------------------------+
//|                                                Demo_FileReadLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Volume"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrYellow
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- les paramètres pour la lecture des données
input string InpFileName="Volume.bin"; // le nom du fichier
input string InpDirectoryName="Data";  // le nom du répertoire
//--- les variables globales
int      ind=0;
int      size=0;
long     volume_buff[];
datetime time_buff[];
//--- indicator buffers
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- Lisons d'abord combien de données sont dans le fichier
      size=(int)FileReadLong(file_handle);
      //--- allouons la mémoire pour les tableaux
      ArrayResize(volume_buff,size);
      ArrayResize(time_buff,size);
      //--- lisons des données du fichier
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadLong(file_handle);
         volume_buff[i]=FileReadLong(file_handle);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- le rattachement du tableau au tampon d'indicateur avec l'indice 0
   SetIndexBuffer(0,buff,INDICATOR_DATA);
```

```
//----l'établissement des valeurs de l'indicateur qui ne seront pas visibles sur le gr
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      buff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               buff[i]=(double)volume_buff[j];
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Les types entiers, FileReadInteger, FileWriteLong

© 2000-2024, MetaQuotes Ltd.

FileReadNumber

Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type double.

```
double  FileReadNumber(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type double.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteString)

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadNumber.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- les paramètres pour la lecture des données
input string InpFileName="RSI.csv";   // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les tampons d'indicateur
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- les variables du surachat
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- les variables de la survente
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- les variables des tailles des tableaux par défaut
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- allouons la mémoire pour les tableaux
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier : %s\\Files\\",TerminalInfoString(TERMINAL_
      double value;
      //--- lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //---lisons la première signification dans la chaîne
         value=FileReadNumber(file_handle);
         //--- lisons aux tableaux différents en fonction du résultat de la fonction
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
```

```
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- le rattachement des tableaux
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
//---- l'établissement des valeurs de l'indicateur, qui ne seront pas vus sur le graph
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| La lecture des données de la chaîne du fichier
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- lisons jusqu'à ce que nous arrivons à la fin d'une chaîne ou d'un fichier
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- rapprocherons le chariot, ayant lu le nombre
      if(flag)
         FileReadNumber(file_handle);
      //--- retenons la date courante
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- augmentons en cas de nécessité la taille du tableau
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- on a passé la première itération
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2024, MetaQuotes Ltd.

```
{
```

© 2000-2024, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- si les dates correspondent, la barre se trouve dans la zone de surac
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - la couleur rouge
               color_buff[i]=0;
               //--- augmentons le compteur
               ovb_ind=j+1;
               break;
              }
           }
      //--- la vérification s'il y a encore des données
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- si les dates correspondent, la barre se trouve dans la zone de la su
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - la couleur bleue
               color_buff[i]=1;
               //---augmentons le compteur
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Le gestionnaire de l'événement ChartEvent                        |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
```

© 2000-2024, MetaQuotes Ltd.

```
                    )
  {
//---changeons l'épaisseur de l'indicateur en fonction de l'échelle
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Voir aussi

FileWriteString

© 2000-2024, MetaQuotes Ltd.

FileReadString

Lit du fichier la chaîne de la position courante de l'indicateur de fichier.

```
string  FileReadString(
   int  file_handle,      // handle du fichier
   int  length=-1         // longueur de la chaîne
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

length=-1

[in] Le nombre de caractères à lire.

La valeur rendue

La ligne lue (string).

Note

A la lecture du fichier bin il est obligatoire d'indiquer la longueur de la chaîne lue. A la lecture du fichier txt il ne faut pas indiquer la longueur de la chaîne, la chaîne sera lue de la position courante jusqu'au caractère de la conversion de la chaîne "\r\n". A la lecture du fichier csv il ne faut pas indiquer la longueur de la chaîne, la chaîne sera lue de la position actuelle jusqu'au délimiteur le plus proche ou jusqu'au caractère de la fin de la chaîne de texte.

Si le fichier est ouvert avec le drapeau FILE_ANSI,la ligne lue est transformée à Unicode.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteInteger)

© 2000-2024, MetaQuotes Ltd.

```
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la lecture des données
input string InpFileName="Trend.bin"; // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- les variables auxiliaires
      int    str_size;
      string str;
      //---lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //---apprenons combien de symboles sont utilisés pour l'inscription du temps
         str_size=FileReadInteger(file_handle,INT_VALUE);
         //--- lisons la chaîne
         str=FileReadString(file_handle,str_size);
         //--- imprimons la chaîne
         PrintFormat(str);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
  }
```

Voir aussi

Le type string, La conversion des données, FileWriteInteger

© 2000-2024, MetaQuotes Ltd.

FileReadStruct

Lit du fichier binaire le contenu à la structure transmise à titre du paramètre. La lecture est produite de la position courante de l'indicateur de fichier.

```
uint  FileReadStruct(
   int          file_handle,        // handle du fichier
   const void&  struct_object,      // structure, où la lecture est faite
   int          size=-1             // dimension de la structure en bytes
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier du fichier binaire ouvert.

struct_object

[out] L'objet de la structure indiquée. La structure ne doit pas contenir les chaînes, les tableaux dynamiques et les fonctions virtuelles.

size=-1

[in] Le nombre de bytes, lesquels il faut lire. Si la dimension n'est pas spécifiée ou le nombre de bytes plus que la taille de la structure est indiqué, on utilise la dimension exacte de la structure indiquée.

La valeur rendue

En cas de succès la fonction rend le nombre de bytes lus. L'indicateur de fichier est déplacé par le même nombre de bytes.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteStruct)

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Candles"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
#property indicator_separate_window
//--- les paramètres pour la réception des données
input string  InpFileName="EURUSD.txt"; // le nom du fichier
input string  InpDirectoryName="Data";  // le nom du répertoire
//+------------------------------------------------------------------+
//| La structure pour la sauvegarde des données de la bougie
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // le prix de l'ouverture
   double            close; // le prix de la clôture
   double            high;  // le prix maximum
   double            low;   // le prix minimum
   datetime          date;  // la date
  };
//--- les tampons d'indicateur
double       open_buff[];
double       close_buff[];
double       high_buff[];
double       low_buff[];
//--- les variables globales
candlesticks cand_buff[];
int          size=0;
int          ind=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int default_size=100;
   ArrayResize(cand_buff,default_size);
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_(
      //--- lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //--- inscrirons les données au tableau
         FileReadStruct(file_handle,cand_buff[size]);
         size++;
```

```
          //--- vérifier un tableau sur l'encombrement
          if(size==default_size)
            {
             //---augmentons la dimension du tableau
             default_size+=100;
             ArrayResize(cand_buff,default_size);
            }
         }
       //--- fermons le fichier
       FileClose(file_handle);
       PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
      }
    else
      {
       PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
       return(INIT_FAILED);
      }
//--- indicator buffers mapping
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
//--- une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
//--- le cycle pour les bougies pas encore traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      open_buff[i]=0;
      close_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      //---la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==cand_buff[j].date)
              {
               open_buff[i]=cand_buff[j].open;
               close_buff[i]=cand_buff[j].close;
               high_buff[i]=cand_buff[j].high;
               low_buff[i]=cand_buff[j].low;
               //--- augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Les structures et les classes, FileWriteStruct

© 2000-2024, MetaQuotes Ltd.

FileSeek

Déplace la position de l'indicateur de fichier par un nombre spécifié de bytes par rapport à la position indiquée.

```
bool  FileSeek(
   int                  file_handle,      // handle du fichier
   long                 offset,           // en bytes
   ENUM_FILE_POSITION   origin            // position pour le décompte
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

offset

[in] Le décalage dans les bytes (peut accepter et la valeur négative).

origin

[in] Le point de départ pour le décalage. Peut accepter une des valeurs de l'énumération ENUM_FILE_POSITION.

La valeur rendue

Rend true en cas du succès, autrement rend false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

Si le résultat de l'exécution de la fonction FileSeek() est le décalage négatif (la sortie pour "une gauche marge" le fichier), l'indicateur de fichier sera établi au début du fichier.

Si exposer la position pour "la marge droite" du fichier (plus que la taille du fichier), l'enregistrement ultérieur au fichier sera produite non de la fin du fichier, mais de la position exposée. En cela entre la fin précédente du fichier et la position exposée on inscrit les valeurs incertaines.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSeek.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="file.txt";    // le nom du fichier
input string InpDirectoryName="Data";   // le nom du répertoire
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- établissons la valeur de la variable pour la génération des nombres accidentels
   _RandomSeed=GetTickCount();
//--- les variables pour les positions du début des chaînes
   ulong pos[];
   int   size;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrons le fichier
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      //--- recevrons la position du début pour chaque chaîne dans le fichier
      GetStringPositions(file_handle,pos);
      //--- définissons le nombre de chaînes dans le fichier
      size=ArraySize(pos);
      if(!size)
        {
         //--- S'il n'y a pas de chaînes dans le fichier, terminons le travail
         PrintFormat("Le fichier %s est vide!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- choisirons le numéro de chaîne par hasard
      int ind=MathRand()%size;
      //--- déplaçons la position au début de cette chaîne
      if(FileSeek(file_handle,pos[ind],SEEK_SET)==true)
        {
      //--- lisons et imprimons la chaîne avec le numéro ind
         PrintFormat("Le texte de la chaîne avec le numéro %d: \"%s\"",ind,FileReadStr
        }
```

```
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Le fichier %s est fermé",InpFileName);
     }
   else
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
  }
//+------------------------------------------------------------------+
//| La fonction définit les positions du début pour chacune des chaînes dans le fichie
//| les place dans un tableau arr                                    |
//+------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- la taille du tableau par défaut
   int def_size=127;
//--- allouons la mémoire pour le tableau
   ArrayResize(arr,def_size);
//--- le compteur des chaînes
   int i=0;
//--- s'il n'y a pas de la fin du fichier, il y a quand même une chaîne
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // le fichier est vide, sortons
//--- définissons le décalage en octets en fonction de l'encodage
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- trions les chaînes dans le cycle
   while(1)
     {
      //--- lisons la chaîne
      FileReadString(handle);
      //--- la vérification à la fin du fichier
      if(!FileIsEnding(handle))
        {
         //--- retiendrons la position de la chaîne suivante
         arr[i]=FileTell(handle)+shift;
         i++;
         //---augmentons la taille du tableau, s'il est rempli
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // la fin du fichier, sortons
     }
//---établissons la taille véritable du tableau
   ArrayResize(arr,i);
  }
```

© 2000-2024, MetaQuotes Ltd.

FileSize

Rend la taille du fichier en bytes.

```
ulong  FileSize(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type int.

Note

Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input ulong  InpThresholdSize=20;        // la frontière de la taille des fichiers en
input string InpBigFolderName="big";     // le dossier pour les grands fichiers
input string InpSmallFolderName="small"; // le dossier pour les petits fichiers
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // la variable pour la sauvegarde des noms des fichiers
   string   filter="*.csv"; // le filtre pour la recherche des fichiers
   ulong    file_size=0;    // la taille du fichier en octets
   int      size=0;         // le nombre de fichier
//--- imprimons la voie vers le dossier dans lequel on va travailler
   PrintFormat("Travaillons dans le dossier %s\\Files\\",TerminalInfoString(TERMINAL_C
//--- la réception du handle de la recherche dans la racine du dossier total de tous l
   long search_handle=FileFindFirst(filter,file_name,FILE_COMMON);
//--- vérifions si la fonction FileFindFirst() a travaillé avec succès
   if(search_handle!=INVALID_HANDLE)
     {
      //--- dans le cycle nous déplaçons les fichiers en fonction de leur taille
      do
        {
         //--- ouvrons le fichier
         ResetLastError();
         int file_handle=FileOpen(file_name,FILE_READ|FILE_CSV|FILE_COMMON);
         if(file_handle!=INVALID_HANDLE)
           {
            //--- recevrons la taille du fichier
            file_size=FileSize(file_handle);
            //--- fermons le fichier
            FileClose(file_handle);
           }
         else
           {
            PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreu
            continue;
           }
         //--- imprimons la taille du fichier
         PrintFormat("La taille du fichier %s est égal au %d octets",file_name,file_si
         //--- définirons la voie pour le déplacement du fichier
         string path;
         if(file_size>InpThresholdSize*1024)
            path=InpBigFolderName+"//"+file_name;
         else
            path=InpSmallFolderName+"//"+file_name;
         //--- déplaçons le fichier
         ResetLastError();
         if(FileMove(file_name,FILE_COMMON,path,FILE_REWRITE|FILE_COMMON))
            PrintFormat("Le fichier %s est déplacé",file_name);
         else
            PrintFormat("L'erreur, le code = %d",GetLastError());
        }
      while(FileFindNext(search_handle,file_name));
      //--- fermons le handle de la recherche
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

© 2000-2024, MetaQuotes Ltd.

FileTell

Rend la position courante de l'indicateur de fichier du fichier correspondant ouvert.

```
ulong  FileTell(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La position courante du descripteur de fichier dans les bytes du début du fichier.

Note

Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                Demo_FileTell.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="file.txt";    // le nom du fichier
input string InpDirectoryName="Data";   // le nom du répertoire
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- établissons la valeur de la variable pour la génération des nombres accidentels
   _RandomSeed=GetTickCount();
//--- les variables pour les positions du début des chaînes
   ulong pos[];
   int   size;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrons le fichier
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      //--- recevrons la position du début pour chaque chaîne dans le fichier
      GetStringPositions(file_handle,pos);
      //--- définissons le nombre de chaînes dans le fichier
      size=ArraySize(pos);
      if(!size)
        {
         //--- S'il n'y a pas de chaînes dans le fichier, terminons le travail
         PrintFormat("Le fichier %s est vide!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- choisirons le numéro de chaîne par hasard
      int ind=MathRand()%size;
      //--- déplaçons la position au début de cette chaîne
      FileSeek(file_handle,pos[ind],SEEK_SET);
      //--- lisons et imprimons la chaîne avec le numéro ind
      PrintFormat("Le texte de la chaîne avec le numéro %d: \"%s\"",ind,FileReadString
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Le fichier %s est fermé",InpFileName);
     }
   else
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
  }
//+------------------------------------------------------------------+
//| La fonction définit les positions du début pour chacune des chaînes dans le fichie
//| les place dans un tableau arr                                    |
//+------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
```

```
//--- la taille du tableau par défaut
   int def_size=127;
//--- allouons la mémoire pour le tableau
   ArrayResize(arr,def_size);
//--- le compteur des chaînes
   int i=0;
//--- s'il n'y a pas de la fin du fichier, il y a quand même une chaîne
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // le fichier est vide, sortons
//--- définissons le décalage en octets en fonction de l'encodage
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- trions les chaînes dans le cycle
   while(1)
     {
      //--- lisons la chaîne
      FileReadString(handle);
      //--- la vérification à la fin du fichier
      if(!FileIsEnding(handle))
        {
         //--- retiendrons la position de la chaîne suivante
         arr[i]=FileTell(handle)+shift;
         i++;
         //---augmentons la taille du tableau, s'il est rempli
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // la fin du fichier, sortons
     }
//---établissons la taille véritable du tableau
   ArrayResize(arr,i);
  }
```

© 2000-2024, MetaQuotes Ltd.

FileWrite

Inscrit les données au fichier comme CSV ou TXT, le délimiteur entre les données est inséré automatiquement, s'il n'est pas égal à 0. Après avoir écrit dans le fichier, le caractère de la fin de la ligne "\r\n" sera ajouté.

```
uint  FileWrite(
   int  file_handle,      // handle du fichier
   ...                    // liste des paramètres inscrits
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

...

[in] La liste des paramètres divisés par les virgules, pour l'enregistrement au fichier. Le nombre de paramètres entrés au fichier ne doit pas excéder 63.

La valeur rendue

Le nombre de bytes écrits.

Note

A la sortie les données numériques seront transformées au format de texte (regarde la fonction Print()). Les données du type double sont sorties à près de 16 chiffres décimaux après le point, en cela les données peuvent être sortis aux formats traditionnel ou scientifique - en fonction de que l'enregistrement sera plus compact. Les données du type float sont sorties avec 5 chiffres décimaux après le point. Pour la sortie des nombres matériels avec une autre exactitude ou dans le format évidemment indiqué il est nécessaire d'utiliser la fonction DoubleToString().

Les nombres du type bool sont sorties en forme des lignes "true" ou "false". Les nombres du type datetime sont sorties au format "YYYY.MM.DD HH:MI:SS".

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                Demo_FileWrite.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string             InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // le temps trame
input int                InpFastEMAPeriod=12;              // la période de la moyenne
input int                InpSlowEMAPeriod=26;              // la période de la moyenne
input int                InpSignalPeriod=9;                // la période de la prise d
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // le type du prix
input datetime           InpDateStart=D'2012.01.01 00:00'; // la date de commencement
//--- les paramètres pour l'enregistrement des données au fichier
input string             InpFileName="MACD.csv";  // le nom du fichier
input string             InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // la date de la fin du copiage des données
   bool     sign_buff[]; // le tableau des signaux (true - l'achat, false - la vente)
   datetime time_buff[]; // le tableau du temps de l'arrivée des signaux
   int      sign_size=0; // la taille des tableaux des signaux
   double   macd_buff[]; // le tableau des valeurs de l'indicateur
   datetime date_buff[]; // le tableau des dates de l'indicateur
   int      macd_size=0; // la taille des tableaux de l'indicateur
//--- le temps de la fin - courant
   date_finish=TimeCurrent();
//--- recevrons le handle de l'indicateur MACD
   ResetLastError();
   int macd_handle=iMACD(InpSymbolName,InpSymbolPeriod,InpFastEMAPeriod,InpSlowEMAPeri
   if(macd_handle==INVALID_HANDLE)
     {
      //--- on n'a pas réussi de recevoir le hadle de l'indicateur
      PrintFormat("L'erreur de la réception du handle de l'indicateur. Le code de l'er
      return;
     }
//--- on se trouve dans le cycle, pendant que l'indicateur ne compte pas toutes les va
   while(BarsCalculated(macd_handle)==-1)
      Sleep(10); // le retard pour que l'indicateur puisse de calculer les valeurs
//---copions les valeurs de l'indicateur pour la période définie
   ResetLastError();
   if(CopyBuffer(macd_handle,0,InpDateStart,date_finish,macd_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs de l'indicateur. Le code de
      return;
     }
//--- copions le temps correspondant pour les valeurs de l'indicateur
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps Le code de l'erreu
      return;
     }
```

```
//--- désallouons la mémoire occupée par l'indicateur
   IndicatorRelease(macd_handle);
//---recevrons la taille du tampon
   macd_size=ArraySize(macd_buff);
//--- analyserons les données et sauvegardons les signaux de l'indicateur aux tableaux
   ArrayResize(sign_buff,macd_size-1);
   ArrayResize(time_buff,macd_size-1);
   for(int i=1;i<macd_size;i++)
     {
      //--- le signal à l'achat
      if(macd_buff[i-1]<0 && macd_buff[i]>=0)
        {
         sign_buff[sign_size]=true;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
      //--- le signal à la vente
      if(macd_buff[i-1]>0 && macd_buff[i]<=0)
        {
         sign_buff[sign_size]=false;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
     }
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- d'abord enregistrons la quantité de signaux
      FileWrite(file_handle,sign_size);
      //--- enregistrons le temps des signaux et leurs valeurs au fichier
      for(int i=0;i<sign_size;i++)
         FileWrite(file_handle,time_buff[i],sign_buff[i]);
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont enregistrés, le fichier %s est fermé",InpFileName)
     }
   else
      PrintFormat("On n'a pas réussi d'ouvrir le fichier %s, Le code de l'erreur = %d'
  }
```

Voir aussi

Comment, Print, StringFormat

© 2000-2024, MetaQuotes Ltd.

FileWriteArray

Inscrit au fichier du type BIN les tableaux des tous les types, sauf les tableaux de chaîne (peut être un tableau des structures ne contenant pas les chaînes et les tableaux dynamiques).

```
uint  FileWriteArray(
   int          file_handle,         // handle du fichier
   const void&  array[],             // tableau
   int          start=0,             // index initial dans le tableau
   int          count=WHOLE_ARRAY    // nombre d'éléments
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

array[]

[out] Le tableau pour l'enregistrement.

start=0

[in] L'index initial dans le tableau (le numéro du premier élément inscrit).

count=WHOLE_ARRAY

[in] Le nombre d'éléments inscrits (WHOLE_ARRAY signifie que s'inscrivent tous les éléments à partir du numéro start jusqu'à la fin du tableau).

La valeur rendue

Le nombre d'éléments inscrits.

Note

Le tableau de chaîne peut s'inscrire seulement au fichier du type TXT. Dans ce cas les chaînes s'achèvent automatiquement par les caractères de la fin de la ligne "\r\n". En fonction du type du fichier ANSI ou UNICODE, les chaînes se transforment au codage ansi, ou non.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteArray.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- les paramètres d'entrée
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| La structure pour stocker des données sur les prix               |
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // lq date
   double            bid;  // le prix bid
   double            ask;  // le prix ask
  };
//--- les variables globales
int    count=0;
int    size=20;
string path=InpDirectoryName+"//"+InpFileName;
prices arr[];
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- l'allocation de la mémoire pour un tableau
   ArrayResize(arr,size);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- un enregistrement des chaînes count restées, si count<n
   WriteData(count);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- sauvegardons les données au tableau
   arr[count].date=TimeCurrent();
   arr[count].bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   arr[count].ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- affichons les données courantes
   Print("Date = ",arr[count].date," Bid = ",arr[count].bid," Ask = ",arr[count].ask);
//---augmentons le compteur
   count++;
//--- si le tableau est rempli, enregistrons les données dans un fichier, et le remett
   if(count==size)
     {
      WriteData(size);
      count=0;
```

```
      }
   }
//+------------------------------------------------------------------+
//| L'enregistrement n des éléments d'un tableau dans un fichier
//+------------------------------------------------------------------+
void WriteData(const int n)
   {
//--- ouvrons le fichier
    ResetLastError();
    int handle=FileOpen(path,FILE_READ|FILE_WRITE|FILE_BIN);
    if(handle!=INVALID_HANDLE)
      {
       //--- enregistrons les données du tableau à la fin du fichier
       FileSeek(handle,0,SEEK_END);
       FileWriteArray(handle,arr,0,n);
       //--- fermons le fichier
       FileClose(handle);
      }
    else
       Print("Failed to open the file, error ",GetLastError());
   }
```

Voir aussi

Les variables, FileSeek

© 2000-2024, MetaQuotes Ltd.

FileWriteDouble

Inscrit au fichier bin la valeur du type double de la position courante de l'indicateur de fichier.

```
uint  FileWriteDouble(
   int     file_handle,     // handle du fichier
   double  value            // valeur pour l'enregistrement
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

value

[in] La valeur du type double.

La valeur rendue

En cas du succès la fonction rend le nombre inscrit des bytes (dans ce cas sizeof(double)=8). L'indicateur de fichier se déplace sur le même nombre de bytes.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//---  montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string             InpSymbolName="EURJPY";           // la paire de devise
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_M15;       // le temps trame
input int                InpMAPeriod=10;                   // la période du lissage
input int                InpMAShift=0;                     // le déplacement de l'indi
input ENUM_MA_METHOD     InpMAMethod=MODE_SMA;             // le type du lissage
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // le type du prix
input datetime           InpDateStart=D'2013.01.01 00:00'; // la date de commencement
//--- les paramètres pour l'enregistrement des données au fichier
input string             InpFileName="MA.csv";    // le nom du fichier
input string             InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   ma_buff[];
   datetime time_buff[];
   int      size;
//--- recevrons le handle de l'indicateur MA
   ResetLastError();
   int ma_handle=iMA(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpMAShift,InpMAMethod,
   if(ma_handle==INVALID_HANDLE)
     {
      //--- on n'a pas réussi de recevoir le hadle de l'indicateur
      PrintFormat("L'erreur de la réception du handle de l'indicateur. Le code de l'er
      return;
     }
//--- on se trouve dans le cycle, pendant que l'indicateur ne compte pas toutes ses va
   while(BarsCalculated(ma_handle)==-1)
      Sleep(20); // le retard pour que l'indicateur puisse de calculer ses valeurs
   PrintFormat("Les valeurs de l'indicateur seront enregistrées au fichier, à partir d
//--- copions les valeurs de l'indicateur
   ResetLastError();
   if(CopyBuffer(ma_handle,0,InpDateStart,date_finish,ma_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs de l'indicateur. Le code de
      return;
     }
//--- copions le temps de l'apparition des barres correspondantes
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//---recevrons la taille du tampon
   size=ArraySize(ma_buff);
//--- désallouons la mémoire occupée par l'indicateur
   IndicatorRelease(ma_handle);
```

```
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- d'abord enregistrons la taille de l'extrait des données
      FileWriteDouble(file_handle,(double)size);
      //--- enregistrons le temps et les valeurs de l'indicateur dans le fichier
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteDouble(file_handle,ma_buff[i]);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sonr enregistrés, le fichier %s est fermé",InpFileName)
     }
   else
      PrintFormat("On n'a pas réussi d'ouvrir le fichier %s, Le code de l'erreur = %d'
  }
```

Voir aussi

Les types entiers (double, float)

© 2000-2024, MetaQuotes Ltd.

FileWriteFloat

Inscrit au fichier bin la valeur du paramètre du type float de la position courante de l'indicateur de fichier.

```
uint  FileWriteFloat(
   int    file_handle,      // handle du fichier
   float  value             // valeur enregistrée
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

value

[in] La valeur du type float.

La valeur rendue

En cas du succès la fonction rend le nombre inscrit des bytes (dans ce cas sizeof(float)=4). L'indicateur de fichier se déplace sur le même nombre de bytes.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string          InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_M15;       // le temps trame
input datetime        InpDateStart=D'2013.01.01 00:00'; // la date de commencement du
//--- les paramètres pour l'enregistrement des données au fichier
input string          InpFileName="Close.bin"; // le nom du fichier
input string          InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- copions le prix de la clôture pour chaque barre
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("On n'a pas réussi de copier les valeurs des prix de la clôture. Le
      return;
     }
//---copions le temps pour chaque barre
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//--- recevrons la taille du tampon
   size=ArraySize(close_buff);
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- enregistrons le temps et les valeurs des prix de la clôture au fichier
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteFloat(file_handle,(float)close_buff[i]);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont enregistrés, le fichier %s est fermé",InpFileName)
     }
   else
      PrintFormat("On n'a pas réussi d'ouvrir le fichier %s, Le code de l'erreur = %d'
```

```
}
```

Voir aussi

Les types entiers (double, float), FileWriteDouble

© 2000-2024, MetaQuotes Ltd.

FileWriteInteger

Inscrit au fichier bin la valeur du paramètre du type int de la position courante de l'indicateur de fichier.

```
uint  FileWriteInteger(
   int  file_handle,         // handle du fichier
   int  value,               // valeur enregistrée
   int  size=INT_VALUE       // taille en bytes
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

value

[in] La valeur entière.

size=INT_VALUE

[in] Le nombre de bytes (jusqu'à 4 y compris), qu'il faut enregistrer. On prévoit les constantes correspondantes: CHAR_VALUE=1, SHORT_VALUE=2 et INT_VALUE=4, ainsi la fonction peut enregistrer la valeur entière du type char, uchar, short, ushort, int ou uint

La valeur rendue

En cas du succès la fonction rend le nombre inscrit des bytes. L'indicateur de fichier se déplace sur le même nombre de bytes.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string             InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // le temps trame
input datetime           InpDateStart=D'2013.01.01 00:00'; // la date de commencement
//--- les paramètres pour l'enregistrement des données au fichier
input string             InpFileName="Trend.bin"; // le nom du fichier
input string             InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//---oblitérons la valeur de l'erreur
   ResetLastError();
//--- copions le prix de la clôture pour chaque barre
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("On n'a pas réussi de copier les valeurs des prix de la clôture. Le
      return;
     }
//---copions le temps pour chaque barre
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//--- recevrons la taille du tampon
   size=ArraySize(close_buff);
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("Le voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //---
      int   up_down=0; // le drapeau de la tendance
      int   arr_size;  // la taille du tableau arr
      uchar arr[];     // le tableau du type uchar
      //--- enregistrons les valeurs du temps au fichier
      for(int i=0;i<size-1;i++)
        {
         //--- comparerons les prix de la clôture des barres courante et suivante
         if(close_buff[i]<=close_buff[i+1])
           {
            if(up_down!=1)
              {
```

```
                //--- enregistrons la valeur de la date au fichier, en utilisant FileWr
                StringToCharArray(TimeToString(time_buff[i]),arr);
                arr_size=ArraySize(arr);
                //--- d'abord enregistrons le nombre de symboles dans le tableau
                FileWriteInteger(file_handle,arr_size,INT_VALUE);
                //--- enregistrons les symboles eux-mêmes
                for(int j=0;j<arr_size;j++)
                   FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
                //--- changeons le drapeau de la tendance
                up_down=1;
               }
            }
         else
            {
             if(up_down!=-1)
               {
                //--- enregistrons la valeur de la date au fichier, en utilisant FileWr
                StringToCharArray(TimeToString(time_buff[i]),arr);
                arr_size=ArraySize(arr);
                //---d'abord enregistrons le nombre de symboles dans le tableau
                FileWriteInteger(file_handle,arr_size,INT_VALUE);
                //--- enregistrons les symboles eux-mêmes
                for(int j=0;j<arr_size;j++)
                   FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
                //--- changeons le drapeau de la tendance
                up_down=-1;
               }
            }
         }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont enregistrés, le fichier %s est fermé",InpFileName)
     }
   else
      PrintFormat("On n'a pas réussi d'ouvrir le fichier %s, Le code de l'erreur = %d"
  }
```

Voir aussi

IntegerToString, StringToInteger, Les types entiers

© 2000-2024, MetaQuotes Ltd.

FileWriteLong

Inscrit au fichier bin la valeur du paramètre du type long de la position courante de l'indicateur de fichier.

```
uint  FileWriteLong(
   int   file_handle,      // handle du fichier
   long  value             // valeur enregistrée
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

value

[in] La valeur du type long.

La valeur rendue

En cas du succès la fonction rend le nombre inscrit des bytes (dans ce cas sizeof(long)=8). L'indicateur de fichier se déplace sur le même nombre de bytes.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string          InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // le temps trame
input datetime        InpDateStart=D'2013.01.01 00:00'; // la date de commencement du
//--- les paramètres pour l'enregistrement des données au fichier
input string          InpFileName="Volume.bin"; // le nom du fichier
input string          InpDirectoryName="Data";  // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   long     volume_buff[];
   datetime time_buff[];
   int      size;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- copions les volumes de tick pour chaque barre
   if(CopyTickVolume(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,volume_buf
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du volume de tick. Le code
      return;
     }
//---copions le temps pour chaque barre
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//--- recevrons la taille du tampon
   size=ArraySize(volume_buff);
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- d'abord enregistrons la taille de l'extrait des données
      FileWriteLong(file_handle,(long)size);
      //--- enregistrons le temps et les valeurs du volume au fichier
      for(int i=0;i<size;i++)
        {
         FileWriteLong(file_handle,(long)time_buff[i]);
         FileWriteLong(file_handle,volume_buff[i]);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont enregistrés, le fichier %s est fermé",InpFileName)
     }
```

```
   else
      PrintFormat("On n'a pas réussi d'ouvrir le fichier %s, Le code de l'erreur = %d'
}
```

Voir aussi

Les types entiers, FileWriteInteger

© 2000-2024, MetaQuotes Ltd.

FileWriteString

Inscrit au fichier comme BIN, CSV ou TXT la valeur du paramètre du type string de la position courante du pointeur de fichier. A l'enregistrement au fichier comme CSV ou TXT, si dans la ligne il y a un symbole '\n' (LF) sans symbole précédant '\r' (CR), devant le symbole '\n' on écrit le symbole manquant '\r'.

```
uint  FileWriteString(
   int           file_handle,     // handle du fichier
   const string  text_string,     // chaîne enregistrée
   int           length=-1        // nombre de symboles
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

text_string

[in] La chaîne.

length=-1

[in] Le nombre de caractères, qu'il faut enregistrer. Le paramètre est nécessaire à l'enregistrement de la chaîne au fichier du type BIN. Si la taille n'est pas indiqué, s'inscrit toute la chaîne sans le 0 terminant. Si la taille indiquée est plus petite que la longueur de la chaîne, on s'inscrit toute la chaîne sans le 0 terminant. Si on indique la taille plus que la longueur de la chaîne, la chaîne s'enregistre par le nombre correspondant de zéros. Pour les fichiers comme CSV et TXT ce paramètre est ignoré et la ligne est enregistrée entièrement.

La valeur rendue

En cas du succès la fonction rend le nombre inscrit des bytes. L'indicateur de fichier se déplace sur le même nombre de bytes.

Note

Il faut avoir en vue que pendant l'enregistrement au fichier ouvert avec le drapeau FILE_UNICODE (ou sans le drapeau FILE_ANSI), le nombre inscrit de bytes sera au 2 fois plus de nombre de caractères inscrits de la chaîne. A l'enregistrement au fichier ouvert avec le drapeau FILE_ANSI, le nombre de bytes enregistrés coïncidera avec le nombre de caractères inscrits de la chaîne.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteString.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string             InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // le temps trame
input int                InpMAPeriod=14;                   // la période de la moyenne
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // le type du prix
input datetime           InpDateStart=D'2013.01.01 00:00'; // la date du commencement
//--- les paramètres pour l'enregistrement des données au fichier
input string             InpFileName="RSI.csv";   // le nom du fichier
input string             InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // la date de la fin du copiage des données
   double   rsi_buff[];  // le tableau des valeurs de l'indicateur
   datetime date_buff[]; // le tableau des dates de l'indicateur
   int      rsi_size=0;  // la taille des tableaux de l'indicateur
//--- le temps de la fin - courant
   date_finish=TimeCurrent();
//--- recevrons le handle de l'indicateur RSI
   ResetLastError();
   int rsi_handle=iRSI(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- on n'a pas réussi de recevoir le hadle de l'indicateur
      PrintFormat("L'erreur de la réception du handle de l'indicateur. Le code de l'er
      return;
     }
//---on se trouve dans le cycle, pendant que l'indicateur ne compte pas toutes ses val
   while(BarsCalculated(rsi_handle)==-1)
      Sleep(10); // le retard pour que l'indicateur puisse de calculer ses valeurs
//--- copions les valeurs de l'indicateur pour la période définie
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,date_finish,rsi_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs de l'indicateur. Le code de
      return;
     }
//--- copions le temps correspondant pour les valeurs de l'indicateur
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//---désallouons la mémoire occupée par l'indicateur
   IndicatorRelease(rsi_handle);
//--- recevrons la taille du tampon
   rsi_size=ArraySize(rsi_buff);
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
```

```
  ResetLastError();
  int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
  if(file_handle!=INVALID_HANDLE)
    {
     PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
     PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
     //--- préparerons les variables auxiliaires
     string str="";
     bool   is_formed=false;
     //--- écrirons les dates sur les zones de surachat et de survente
     for(int i=0;i<rsi_size;i++)
       {
        //--- la vérification des valeurs de l'indicateur
        if(rsi_buff[i]>=70 || rsi_buff[i]<=30)
          {
           //--- si c'est la première valeur dans cette zone
           if(!is_formed)
             {
              //--- ajoutons la valeur et la date
              str=(string)rsi_buff[i]+"\t"+(string)date_buff[i];
              is_formed=true;
             }
           else
              str+="\t"+(string)rsi_buff[i]+"\t"+(string)date_buff[i];
           //--- le passage à l'itération suivante du cycle
           continue;
          }
        //--- la vérification du drapeau
        if(is_formed)
          {
           //---la ligne est formée, enregistrons au fichier
           FileWriteString(file_handle,str+"\r\n");
           is_formed=false;
          }
       }
     //--- fermons le fichier
     FileClose(file_handle);
     PrintFormat("Les données sont enregistrés, le fichier %s est fermé",InpFileName)
    }
  else
     PrintFormat("Les données sont inscrites, le fichier %s est fermé = %d",InpFileNa
}
```

Voir aussi

Le type string, StringFormat

© 2000-2024, MetaQuotes Ltd.

FileWriteStruct

Inscrit au fichier bin le contenu de la structure, transmise à titre du paramètre de la position courante de l'indicateur de fichier.

```
uint  FileWriteStruct(
   int          file_handle,       // handle du fichier
   const void&  struct_object,     // lien à l'objet
   int          size=-1            // taille pour enregistrement en bytes
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

struct_object

[in] Le lien à l'objet de la structure indiquée. La structure ne doit pas contenir les chaînes, les tableaux dynamiques et les fonctions virtuelles.

size=-1

[in] Le nombre de bytes, lesquels il faut lire. Si la dimension n'est pas spécifiée ou le nombre de bytes plus que la taille de la structure est indiqué, on enregistre toute la structure.

La valeur rendue

En cas de succès la fonction rend le nombre de bytes lus. L'indicateur de fichier est déplacé par le même nombre de bytes.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWiteStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string          InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // le temps trame
input datetime        InpDateStart=D'2013.01.01 00:00'; // la date de commencement du
//--- les paramètres pour l'enregistrement des données au fichier
input string          InpFileName="EURUSD.txt";         // le nom du fichier
input string          InpDirectoryName="Data";          // le nom du répertoire
//+------------------------------------------------------------------+
//| La structure pour stocker des données de la bougie
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // le prix de l'ouverture
   double            close; // le prix de la clôture
   double            high;  // le prix maximum
   double            low;   // le prix minimum
   datetime          date;  // la date
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime     date_finish=TimeCurrent();
   int          size;
   datetime     time_buff[];
   double       open_buff[];
   double       close_buff[];
   double       high_buff[];
   double       low_buff[];
   candlesticks cand_buff[];
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons le temps de l'apparition des barres de la gamme
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//--- recevrons les prix maximals des barres de la gamme
   if(CopyHigh(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,high_buff)==-1)
     {
      PrintFormat(On n'a pas réussi de copier les valeurs des prix maximales. Le code
      return;
     }
//--- recevrons les prix minimales des barres de la gamme
   if(CopyLow(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,low_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs des prix minimales. Le code
      return;
     }
```

```
//--- recevrons les prix de l'ouverture des barres de la gamme
   if(CopyOpen(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,open_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs des prix de l'ouverture. Le
      return;
     }
//--- recevrons les prix de la clôture des barres de la gamme
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("On n'a pas réussi de copier les valeurs des prix de la clôture. Le
      return;
     }
//--- définirons la dimension des tableaux
   size=ArraySize(time_buff);
//--- sauvegardons toutes les structures données dans le tableau
   ArrayResize(cand_buff,size);
   for(int i=0;i<size;i++)
     {
      cand_buff[i].open=open_buff[i];
      cand_buff[i].close=close_buff[i];
      cand_buff[i].high=high_buff[i];
      cand_buff[i].low=low_buff[i];
      cand_buff[i].date=time_buff[i];
     }

//--- ouvrirons le fichier pour l'enregistrement du tableau de la structure au fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_D
      //--- préparerons le compteur de la quantité des octets
      uint counter=0;
      //---enregistrons les valeurs du tableau dans le cycle
      for(int i=0;i<size;i++)
         counter+=FileWriteStruct(file_handle,cand_buff[i]);
      PrintFormat(" %d octets de l'information sont enregistrés au fichier %s",InpFile
      PrintFormat("Au total octets: %d * %d * %d = %d, %s",size,5,8,size*5*8,size*5*8=
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont enregistrées, le fichier %s est fermé",InpFileNam
     }
   else
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
  }
```

Voir aussi

Les structures et les classes

© 2000-2024, MetaQuotes Ltd.

FileLoad

Lit toutes les données d'un fichier binaire spécifié et remplit le tableau d'éléments numériques ou de structures simples passé en argument. La fonction vous permet de lire rapidement des données d'un type connu et de remplir le tableau correspondant.

```
long  FileLoad(
   const string  file_name,          // Nom du fichier
   void&         buffer[],           // Tableau d'éléments numériques ou de structures
   int           common_flag=0       // Un flag du fichier, recherché dans <data_folde
   );
```

Paramètres

file_name

[in] Le nom du fichier dans lequel les données seront lues.

buffer

[out] Un tableau d'éléments numériques ou de structures simples.

common_flag=0

[in] Un flag du fichier indiquant le mode d'opération. Si le paramètre n'est pas spécifié, le fichier est recherché dans le sous-répertoire MQL5\Files (ou dans <répertoire_de_agent_de_test>\MQL5\Files en cas de test).

Valeur de Retour

Le nombre d'éléments lus ou -1 en cas d'erreur.

Note

La fonction FileLoad() lit depuis un fichier le nombre d'octets multiplié par la taille du tableau. Supposons que la taille du fichier est 10 octets et que la fonction lit les données et remplit un tableau de doubles ((sizeof(double)=8). Dans ce cas, la fonction ne lira que 8 octets, les 2 octets restants à la fin du fichiers seront ignorés et la fonction FileLoad() retournera 1 (1 élément lu).

Exemple :

```
//+------------------------------------------------------------------+
//|                                                   Demo_FileLoad.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- paramètres d'entrée
input int      bars_to_save=10; // Nombre de barres
//+------------------------------------------------------------------+
```

```
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_rates.bin";
   MqlRates rates[];
//---
   int copied=CopyRates(_Symbol,_Period,0,bars_to_save,rates);
   if(copied!=-1)
     {
      PrintFormat(" CopyRates(%s) a copié %d barres",_Symbol,copied);
      //---  Ecriture des cotations dans un fichier
      if(!FileSave(filename,rates,FILE_COMMON))
         PrintFormat("FileSave() a échoué, erreur=%d",GetLastError());
     }
   else
      PrintFormat("Echec de CopyRates(%s), erreur=",_Symbol,GetLastError());
//--- Relecture des cotations depuis le fichier
   ArrayFree(rates);
   long count=FileLoad(filename,rates,FILE_COMMON);
   if(count!=-1)
     {
      Print("Time\tOpen\tHigh\tLow\tClose\tTick Voulme\tSpread\tReal Volume");
      for(int i=0;i<count;i++)
        {
         PrintFormat("%s\t%G\t%G\t%G\t%G\t%I64u\t%d\t%I64u",
                     TimeToString(rates[i].time,TIME_DATE|TIME_SECONDS),
                     rates[i].open,rates[i].high,rates[i].low,rates[i].close,
                     rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
  }
```

Voir aussi

Structures et Classes, FileReadArray, FileReadStruct, FileSave

© 2000-2024, MetaQuotes Ltd.

FileSave

Ecrit tous les éléments d'un tableau passé en argument dans un fichier binaire. La fonction vous permet d'écrire rapidement des tableaux d'élements numériques ou de structures simples sous forme d'une chaîne de caractères.

```
bool  FileSave(
   const string  file_name,          // Nom du fichier
   void&         buffer[],           // Tableau d'éléments numériques ou de structures
   int           common_flag=0       // Un flag de fichier, par default, les fichiers s
   );
```

Paramètres

file_name

[in] Le nom du fichier dans lequel le tableau de données sera écrit.

buffer

[in] Un tableau d'éléments numériques ou de structures simples.

common_flag=0

[in] Un flag du fichier indiquant le mode d'opération. Si le paramètre n'est pas spécifié, le fichier sera écrite dans le sous-répertoire MQL5\Files (ou dans <repertoire_agent_de_test>\MQL5\Files en cas de test).

Valeur de Retour

En cas d'erreur, retourne false.

Exemple :

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSave.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- paramètres d'entrée
input int      ticks_to_save=1000; // Nombre de ticks
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_ticks.bin";
   MqlTick ticks[];
//---
   int copied=CopyTicks(_Symbol,ticks,COPY_TICKS_ALL,0,ticks_to_save);
   if(copied!=-1)
```

```
      {
       PrintFormat(" CopyTicks(%s) a copié %d ticks",_Symbol,copied);
       //--- Si l'historique des ticks esst synchronisé, le code d'erreur est égal à zé
       if(!GetLastError()==0)
          PrintFormat("%s : Les ticks ne sont pas synchronisés, erreur=%d",_Symbol,copi
       //---  Ecriture des ticks dans un fichier
       if(!FileSave(filename,ticks,FILE_COMMON))
          PrintFormat("FileSave() a échoué, erreur=%d",GetLastError());
      }
    else
       PrintFormat("Echec de CopyTicks(%s), Erreur=",_Symbol,GetLastError());
//--- Relecture des ticks depuis le fichier
    ArrayFree(ticks);
    long count=FileLoad(filename,ticks,FILE_COMMON);
    if(count!=-1)
      {
       Print("Time\tBid\tAsk\tLast\tVolume\tms\tflags");
       for(int i=0;i<count;i++)
         {
          PrintFormat("%s.%03I64u:\t%G\t%G\t%G\t%I64u\t0x%04x",
          TimeToString(ticks[i].time,TIME_DATE|TIME_SECONDS),ticks[i].time_msc%1000,
          ticks[i].bid,ticks[i].ask,ticks[i].last,ticks[i].volume,ticks[i].flags);
         }
      }
   }
```

Voir aussi

Structures et Classes, FileWriteArray, FileWriteStruct, FileLoad, FileWrite

© 2000-2024, MetaQuotes Ltd.

FolderCreate

Crée le répertoire dans le dossier Files (en fonction de la valeur common_flag)

```
bool  FolderCreate(
   string  folder_name,        // chaîne avec le nom du dossier créé
   int     common_flag=0       // domaine d'action
   );
```

Paramètres

folder_name

[in] Le nom du répertoire, qu'il faut créer. Contains the relative path to the folder.

common_flag=0

[in] Le drapeau, définissant la situation du répertoire. Si common_flag=FILE_COMMON, le répertoire se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le répertoire se trouve dans le dossier local (MQL5\Files ou MQL5\Tester\Files en cas du test).

La valeur rendue

Rend true en cas du succès, autrement rend false.

Exemple:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- description
#property description "The script shows FolderCreate() application sample."
#property description "The external parameter defines the directory for creating folde
#property description "The folder structure is created after executing the script"

//--- display window of the input parameters during the script's launch
#property script_show_inputs
//--- the input parameter defines the folder, in which the script works
input bool     common_folder=false; // common folder for all terminals
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- folder to be created in MQL5\Files
   string root_folder="Folder_A";
   if(CreateFolder(root_folder,common_folder))
     {
      //--- create the Child_Folder_B1 sub-folder in it
      string folder_B1="Child_Folder_B1";
      string path=root_folder+"\\"+folder_B1;          // create the folder name consi
      if(CreateFolder(path,common_folder))
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- create 3 more sub-directories in this folder
         string folder_C11="Child_Folder_C11";
         string child_path=root_folder+"\\"+folder_C11;// create the folder name consi
         CreateFolder(child_path,common_folder);
         //--- second sub-directory
         string folder_C12="Child_Folder_C12";
         child_path=root_folder+"\\"+folder_C12;
         CreateFolder(child_path,common_folder);

         //--- third sub-directory
         string folder_C13="Child_Folder_C13";
         child_path=root_folder+"\\"+folder_C13;
         CreateFolder(child_path,common_folder);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Try creating a folder and display a message about that           |
//+------------------------------------------------------------------+
bool CreateFolder(string folder_path,bool common_flag)
  {
   int flag=common_flag?FILE_COMMON:0;
   string working_folder;
//--- define the full path depending on the common_flag parameter
   if(common_flag)
      working_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH)+"\\MQL5\\Files";
   else
      working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- debugging message
   PrintFormat("folder_path=%s",folder_path);
//--- attempt to create a folder relative to the MQL5\Files path
   if(FolderCreate(folder_path,flag))
     {
      //--- display the full path for the created folder
      PrintFormat("Created the folder %s",working_folder+"\\"+folder_path);
      //--- reset the error code
      ResetLastError();
      //--- successful execution
      return true;
     }
   else
      PrintFormat("Failed to create the folder %s. Error code %d",working_folder+folde
//--- execution failed
   return false;
  }
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

FileOpen(), FolderClean(), FileCopy()

© 2000-2024, MetaQuotes Ltd.

FolderDelete

Supprime le répertoire indiqué. Si le dossier n'est pas vide, elle ne peut pas être supprimé.

```
bool  FolderDelete(
   string  folder_name,        // chaîne avec le nom du dossier supprimé
   int     common_flag=0       // domaine d'action
   );
```

Paramètres

folder_name

[in] Le nom du répertoire, qu'il faut supprimer. Contient le chemin complet vers le dossier.

common_flag=0

[in] Le drapeau, définissant la situation du répertoire. Si common_flag=FILE_COMMON, le répertoire se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le répertoire se trouve dans le dossier local (MQL5\files ou MQL5\tester\files en cas du test).

La valeur rendue

Rend true en cas du succès, autrement rend false.

Note

Si le répertoire contient au moins un fichier et/ou le sous-répertoire, l'effacement d'un tel répertoire est impossible, il est nécessaire préalablement de l'effacer. Le nettoyage total du dossier de tous les fichiers et les sous-dossiers inclus s'effectue à l'aide de la fonction FolderClean().

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          Demo_FolderDelete.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- la description
#property description "Le script montre un exemple de l'utilisation FolderDelete()."
#property description "D'abord, deux dossiers sont créés, l'un est vide et l'autre co
#property description "Si vous essayez de supprimer un dossier non vide vous recevrez

//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string   firstFolder="empty";    // un dossier vide
input string   secondFolder="nonempty";// le dossier, dans lequel il y aura un fichie
string filename="delete_me.txt";       // le nom du fichier que nous créons dans le d
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- écrivons ici le handle du fichier
   int handle;
//---savons dans quel dossier nous travaillons
   string working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- le message de débogage
   PrintFormat("working_folder=%s",working_folder);
//--- la tentative de créer le dossier vide par rapport au chemin MQL5\Files
   if(FolderCreate(firstFolder,0)) // 0 signifie que nous travaillons dans un dossier
     {
      //--- déduisons un chemin complet jusqu'au dossier créé
      PrintFormat("On a créé le dossier %s",working_folder+"\\"+firstFolder);
      //--- oblitérons le code de l'erreur
      ResetLastError();
     }
   else
      PrintFormat("On n'a pas réussi à créer le dossier %s. Le code de l'erreur %d",w

//--- maintenant nous créons le dossier non vide à l'aide de la fonction FileOpen()
   string filepath=secondFolder+"\\"+filename;  // formeront le chemin vers le fichie
   handle=FileOpen(filepath,FILE_WRITE|FILE_TXT); // le drapeau FILE_WRITE dans ce ca
   if(handle!=INVALID_HANDLE)
      PrintFormat("on a ouvert le fichier pour la lecture %s",working_folder+"\\"+fil
   else
      PrintFormat("On n'a pas réussi à créer le fichier %s dans le dossier %s. Le cod

   Comment(StringFormat("préparons de supprimer les dossiers %s et %s", firstFolder, 
//--- Une petite pause à 5 secondes pour que nous puissions lire le message sur le gr
   Sleep(5000); // Sleep() on ne peut pas utiliser dans les indicateurs!

//--- déduisons la boîte de dialogue et nous demandons à l'utilisateur
   int choice=MessageBox(StringFormat("Supprimer les dossiers %s et %s?", firstFolder,
                         "La suppression des dossiers",
                         MB_YESNO|MB_ICONQUESTION); //  Il y aura deux boutons - "Yes

//--- effectuons des actions selon l'option choisie
   if(choice==IDYES)
     {
```

```
      //--- effaçons le commentaire sur le graphique
      Comment("");
      //--- déduisons le message au journal "Experts"
      PrintFormat("Essayons de supprimer les dossiers %s et %s",firstFolder, secondFol
      ResetLastError();
      //--- supprimons un dossier vide
      if(FolderDelete(firstFolder))
         //--- doivent voir ce message, puisque le dossier est vide
         PrintFormat("Le dossier %s est supprimé avec succès",firstFolder);
      else
         PrintFormat("On n'a pas réussi à supprimer le dossier %s. Le code de l'erreur

      ResetLastError();
      //--- supprimons le dossier qui contient le fichier
      if(FolderDelete(secondFolder))
         PrintFormat("Le dossier %s est supprimé avec succès", secondFolder);
      else
         //--- doivent voir ce message, puisque dans le dossier il y a un fichier
         PrintFormat("On n'a pas réussi à supprimer le dossier %s. Le code de l'erreur
     }
   else
      Print("La suppression est annulée");
//---
  }
```

Voir aussi

FileOpen(), FolderClean(), FileMove()

© 2000-2024, MetaQuotes Ltd.

FolderClean

Supprime tous les fichiers dans le dossier indiqué.

```
bool  FolderClean(
   string  folder_name,        // chaîne avec le nom du dossier
   int     common_flag=0       // domaine d'action
   );
```

Paramètres

folder_name

[in] Le nom du répertoire,où il faut supprimer tous les fichiers. Contient le chemin complet vers le dossier.

common_flag=0

[in] Le drapeau, définissant la situation du répertoire. Si common_flag=FILE_COMMON, le répertoire se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le répertoire se trouve dans le dossier local (MQL5\files ou MQL5\tester\files en cas du test).

La valeur rendue

Rend true en cas du succès, autrement rend false.

Note

Se servez prudemment de cette fonction, puisque tous les fichiers et tous les sous-répertoires inclus se suppriment irrévocablement.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              Demo_FolderClean.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- la description
#property description "Le script montre un exemple de l'utilisation FolderClean()."
#property description "D'abord on crée les fichiers dans le dossier indiqué à l'aide d
#property description "Puis, avant la suppression des fichiers on déduit l'avertisseme

//--- Nous montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string   foldername="demo_folder";  // créons le dossier dans MQL5/Files/
input int      files=5;                   // Combien de fichiers nous créons et nous s
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string name="testfile";
//--- d'abord nous ouvrirons ou nous créons les fichiers dans le dossier des données d
   for(int N=0;N<files;N++)
     {
      //--- collectons le nom du fichier comme 'demo_folder\testfileN.txt'
      string filemane=StringFormat("%s\\%s%d.txt",foldername,name,N);
      //--- ouvrons le fichier avec le drapeau sur l'enregistrement, dans ce cas le do
      int handle=FileOpen(filemane,FILE_WRITE);
      //--- connaissons, autant la fonction a été accompli avec succès FileOpen()
      if(handle==INVALID_HANDLE)
        {
         PrintFormat("On n'a pas réussi à créer le fichier %s. Le code de l'erreur ",f
         ResetLastError();
        }
      else
        {
         PrintFormat("Le fichier %s est ouvert avec succès",filemane);
         //--- nous n'avons plus besoin du fichier ouvert, il faut le fermer obligato
         FileClose(handle);
        }
     }

//---vérifions combien de fichiers sont dans le dossier
   int k=FilesInFolder(foldername+"\\*.*",0);
   PrintFormat("Au total%s on a trouvé %d fichiers",foldername,k);

//--- déduisons la boîte de dialogue et demanderons à l'utilisateur
   int choice=MessageBox(StringFormat("Vous allez supprimer %s %d fichiers du dossier,
                         "La suppression des fichiers du dossier",
                         MB_YESNO|MB_ICONQUESTION); //  il y aura deux boutons - "Yes"
   ResetLastError();

//--- accomplirons les actions en fonction de la variante choisie
   if(choice==IDYES)
     {
      //--- commençons à supprimer
      PrintFormat("La tentative de supprimer tous les fichiers du dossier %s",folderna
      if(FolderClean(foldername,0))
```

```
         PrintFormat("Les fichiers sont supprimés avec succès, %s fichiers %d sont rés
                     foldername,
                     FilesInFolder(foldername+"\\*.*",0));
      else
         PrintFormat("On n'a pas réussi à supprimer les fichiers du dossier %s. Le coc
     }
   else
      PrintFormat("La suppression est annulée");
//---
  }
//+------------------------------------------------------------------+
//| rend le nombre de fichiers dans le dossier                       |
//+------------------------------------------------------------------+
int FilesInFolder(string path,int flag)
  {
   int count=0;
   long handle;
   string filename;
//---
   handle=FileFindFirst(path,filename,flag);
//--- Si quand même un fichier est trouvé, nous cherchons les autres
   if(handle!=INVALID_HANDLE)
     {
      //--- déduisons le nom du fichier
      PrintFormat("le fichier %s est trouvé",filename);
      //--- accroissons le compteur des fichiers/dossiers trouvés
      count++;
      //--- commençons le balayage de tous les fichiers/dossiers
      while(FileFindNext(handle,filename))
        {
         PrintFormat("le fichier %s a été trouvé",filename);
         count++;
        }
      //--- fermons obligatoirement le handle de la recherche à la fin
      FileFindClose(handle);
     }
   else // on n'a pas réussi à recevoir le handle
     {
      PrintFormat("On n'a pas réussi à faire la recherche des fichiers dans le dossier
     }
//--- rendons le résultat
   return count;
  }
```

Voir aussi

FileFindFirst, FileFindNext, FileFindClose

© 2000-2024, MetaQuotes Ltd.

Les indicateurs d'utilisateur

Le groupe des fonctions utilisées lors de la présentation des indicateurs d'utilisateur. On ne peut pas utiliser ces fonctions pour écrire les conseillers et les scripts.

Fonction	Action
SetIndexBuffer	Attache le tampon d'indicateur indiquée avec le tableau unidimentionnel dynamique du type double
IndicatorSetDouble	Spécifie la valeur de la propriété d'indicateur, ayant le type double
IndicatorSetInteger	Spécifie la valeur de la propriété d'indicateur, ayant le type int
IndicatorSetString	Spécifie la valeur de la propriété d'indicateur, ayant le type string
PlotIndexSetDouble	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type double
PlotIndexSetInteger	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type int
PlotIndexSetString	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type string
PlotIndexGetInteger	Rend la valeur de la propriété de la ligne d'indicateur, ayant le type entier

Propriétés des indicateurs peuvent être installées à l'aide des directives de la compilation, et à l'aide des fonctions. Pour mieux comprendre il est recommandé d'étudierles styles des indicateurs dans les exemples.

Il est nécessaire de placer tous les comptes nécessaires des indicateurs d'utilisateur dans la fonction prédéterminée OnCalculate(). Si on utilise la forme courte de l'appel de la fonction OnCalculate() de l'aspect suivant

```
int OnCalculate (const int rates_total, const int prev_calculated, const int begin, co
```

la variable *rates_total* contient la valeur du nombre total d'éléments du tableau price[], transmis à titre du paramètre d'entrée pour le calcul des valeurs d'indicateur.

Le paramètre *prev_calculated* - c'est le résultat de l'exécution de la fonction OnCalculate() sur l'appel précédent et permet d'organiser l'algorithme économe du calcul des valeurs d'indicateur. Par exemple, si la valeur courante rates_total=1000, et prev_calculated=999, probablement, il nous suffit de faire les calculs seulement pour une valeur de chaque tampon d'indicateur.

Si l'information sur la grandeur du tableau d'entrée price était inaccessible, cela a amené à la nécessité de faire les calculs pour les 1000 valeurs de chaque tampon d'indicateur. Au premier appel de la fonction OnCalculate() la valeur prev_calculated=0. Si le tableau price[] a changé d'une façon ou d'une autre,dans ce cas prev_calculated est aussi égal au 0.

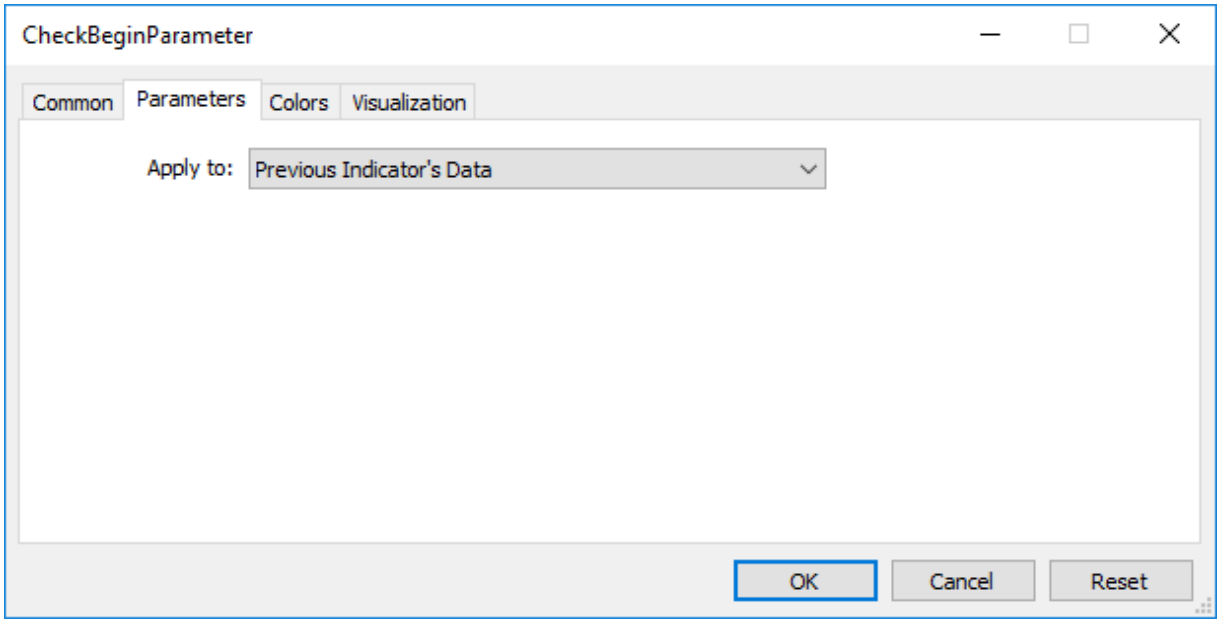
Le paramètre *begin* montre le nombre de valeurs initiales du tableau price, qui ne contient pas de données pour le calcul. Par exemple, si à titre du du tableau d'entrée on utilisait les valeurs de l'indicateur Accelerator Oscillator (pour lesquelles les premiers 37 valeurs ne sont pas calculées), begin=37. Comme l'exemple nous examinerons l'indicateur simple:

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])

  {
//---
   Print("begin = ",begin,"  prev_calculated = ",prev_calculated,"  rates_total = ",ra
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Traînez-le de la fenêtre "Navigateur" à la fenêtre de l'indicateur Accelerator Oscillator et indiquons que les calculs seront produits sur les valeurs de l'indicateur précédent:

© 2000-2024, MetaQuotes Ltd.



A la suite du premier appel de la fonction OnCalculate() la valeur prev_calculated sera égal au zéro, mais aux appels ultérieurs il sera égal à la valeur rates_total (jusqu'à ce que le nombre de barres sur le graphique de prix n'augmentera pas).

La valeur du paramètre begin sera en exactitude égal au nombre de barres initialles, pour lesauelles les valeurs de l'indicateur Accelerator ne sont pas calculées conformément à la logique de cet indicateur. Si nous regardons le code primaire de l'indicateur d'utilisateur Accelerator.mq5, nous verrons dans la fonction OnInit() les telles lignes:

© 2000-2024, MetaQuotes Ltd.

```
//--- sets first bar from what index will be drawn
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,37);
```

Notamment par la fonction PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, empty_first_values)nous spécifions le nombre de premières valeurs irrelevantes au tableau nul d'indicateur de l'indicateur d'utilisateur, lesquelles il ne faut pas prendre en attention pour les calculs (empty_first_values). Ainsi, nous avons des mécanismes pour:

1. définissez le nombre de valeurs de l'indicateur, qui ne devraient pas être utilisé pour les calculs dans un autre indicateur d'utilisateur;
2. recevoir l'information sur le nombre de premières valeurs, qu'il est nécessaire d'ignorer à l'appel de l'autre indicateur d'utilisateur, sans entrer dans la logique de ses calculs.

© 2000-2024, MetaQuotes Ltd.

Styles des indicateurs dans les exemples

Dans le terminal de client MetaTrader 5 il y a 38 indicateurs techniques qui peuvent être utilisés dans les programmes MQL5 à l'aidedes fonctions correspondantes. Mais un principal avantage du langage MQL5 - c'est la possibilité de créer ses propres indicateurs d'utilisateur, qui peuvent ensuite être utilisées dans les conseillers pour recevoir des valeurs ou tout simplement d'imposer sur les graphiques de prix pour l'analyse technique.

On peut recevoir l'ensemble des indicateurs sur la base de quelquesstyles de la dessin, appelés appelé comme les constructions graphiques. La construction est considérée comme le moyen de l'affichage des données, lesquelles l'indicateur calcule, garde et donne selon la demande. Au total il y a sept constructions de base:

1. la ligne,
2. la section (le segment),
3. l'histogramme,
4. la flèche (le symbole)
5. le domaine peint (le canal avec le remplissage),
6. les barres,
7. les chandeliers japonnais.

Chaque construction exige pour son affichage de 1 à cinq tableauxdu type doubleoù se trouvent les valeurs de l'indicateur. Ces tableaux se connectent aux tampons d'indicateur pour le confort du fonctionnement de l'indicateur. Il faut déclarer le nombre de tampons dans l'indicateur à l'aide de ladirective du compilateur, par exemple:

```
#property indicator_buffers 3 // le nombre de tampons
#property indicator_plots   2 // le nombre de constructions graphiques
```

Le nombre de tampons dans l'indicateur est toujours plus ou égal au nombre de constructions dans l'indicateur.

Comme chaque construction de base graphique peut avoir les variations de couleur ou l'affichage spécifique, le nombre réel de constructions dans le langage de MQL5 est 18:

La construction	La description	Les tampons des valeurs	Des tampons de la couleur
DRAW_NONE	N'affiche pas visuellement sur le graphique, mais on peut regarder les valeurs du	1	-

© 2000-2024, MetaQuotes Ltd.

La construction	La description	Les tampons des valeurs	Des tampons de la couleur
	tampon correspondant dans "Fenêtre des données"		
DRAW_LINE	On construit la ligne selon les valeurs du tampon correspondant (les valeurs vides dans le tampon sont indésirables)	1	-
DRAW_SECTION	Se dessine par les segments entre les valeurs du tampon correspondant (d'habitude il y a beaucoup de valeurs vides)	1	-
DRAW_HISTOGRAM	Se dessine par	1	-

© 2000-2024, MetaQuotes Ltd.

La construction	La description	Les tampons des valeurs	Des tampons de la couleur
	l'histogramme de la ligne zéro aux valeurs du tampon correspondant (on admet les valeurs vides)		
DRAW_HISTOGRAM2	Se dessine par l'histogramme sur deux tampons d'indicateur (on admet les valeurs vides)	2	-
DRAW_ARROW	S'affiche par les symboles (on admet les valeurs vides)	1	-
DRAW_ZIGZAG	Est semblable au style DRAW_SECTION, mais à la différen	2	-

© 2000-2024, MetaQuotes Ltd.

La construction	La description	Les tampons des valeurs	Des tampons de la couleur
	ce de lui il peut construire les segments verticaux sur une barre		
DRAW_FILLING	Remplissage de couleur entre les deux lignes. Dans "La fenêtre des données" sont affichées les valeurs des deux tampons correspondants	2	-
DRAW_BARS	L'affichage en forme de barres sur le graphique. Dans "La fenêtre des données" sont montrées les 4 valeurs des	4	-

© 2000-2024, MetaQuotes Ltd.

La construction	La description	Les tampons des valeurs	Des tampons de la couleur
	tampons correspondants		
DRAW_CANDLES	L'affichage en forme de chandeliers japonais. Dans "La fenêtre des données" sont montrées les 4 valeurs des tampons correspondants	4	-
DRAW_COLOR_LINE	La ligne pour laquelle vous pouvez alterner les couleurs sur les barres différentes ainsi que changer sa couleur à n'importe quel moment du temps	1	1

© 2000-2024, MetaQuotes Ltd.

La construction	La description	Les tampons des valeurs	Des tampons de la couleur
DRAW_COLOR_SECTION	Est semblable au styleDRAW_SECTION, mais on peut spécifier la couleur de chaque section individuellement, on peut spécifier la couleur dynamiquement	1	1
DRAW_COLOR_HISTOGRAM	Est semblable au styleDRAW_HISTOGRAM, mais chaque bande peut avoir la couleur, on peut spécifier la couleur dynamiquement	1	1
DRAW_COLOR_HISTOGRAM2	Est semblable au style DRAW_	2	1

© 2000-2024, MetaQuotes Ltd.

La construction	La description	Les tampons des valeurs	Des tampons de la couleur
	HISTOGRAM2, mais chaque bande peut avoir la couleur, on peut spécifier la couleur dynamiquement		
DRAW_COLOR_ARROW	Est semblable au styleDRAW_ARROW, mais chaque symbole peut avoir sa couleur. La couleur peut être changée dynamiquement	1	1
DRAW_COLOR_ZIGZAG	Le styleDRAW_ZIGZAG avec la possibilité du coloriage individuel des sections et le	2	1

© 2000-2024, MetaQuotes Ltd.

La construction	La description	Les tampons des valeurs	Des tampons de la couleur
	remplacement dynamique de la couleur		
DRAW_COLOR_BARS	Le styleDRAW_BARS avec la possibilité du coloriage individuel des barres et le remplacement dynamique de la couleur	4	1
DRAW_COLOR_CANDLES	Le styleDRAW_CANDLES avec la possibilité du coloriage individuel des bougies et le remplacement dynamique de la couleur	4	1

La différence entre le tampon d'indicateur et le tableau

© 2000-2024, MetaQuotes Ltd.

Il faut déclarer dans chaque indicateur auniveau global un ou plusieurs tableaux de type double, qui doit être utilisé après comme un tampon d'indicateur à l'aide de la fonctionSetIndexBuffer(). Pour dessiner les constructions graphiques de l'indicateur sont utilisés seulement les valeurs de tampons d'indicateur, ou on ne peut pas utiliser les autres tableaux pour cela. En outre, les valeurs des tampons sont montrées dans "la Fenêtre des données".

Un tampon d'indicateur doit être dynamique et ne nécessite pas l'indication de la taille - la taille du tableau utilisé comme le tampon d'indicateur, s'installe automatiquement par le sous-système exécutif du terminal.

La direction de l'indexation après le binding du tableau avec le tampon d'indicateur s'établit par défaut comme dans les tableaux ordinaires, mais on peut appliquer en cas de nécessité la fonctionArraySetAsSeries() pour le changement du moyen de l'accès aux éléments du tableau. Par défaut, le tampon d'indicateur est utilisé pour stocker les données destinés pour le dessin (INDICATOR_DATA .)

Si pour les calculs des valeurs de l'indicateur il faut faire les calculs intermédiaires et garder la valeur auxiliaire pour chaque barre, pendant le binding on peut déclarer un tel tableau à titre du tampon de calcul (INDICATOR_CALCULATIONS). On peut utiliser un tableau ordinaire pour des valeurs intermédiaires, mais dans ce cas le programmeur doit gérer la taille de tel tableau.

Certaines constructions permettent de spécifier pour chaque barre la couleur de l'image. Pour stocker l'information sur la couleur on utilise les tampons de couleur (INDICATOR_COLOR_INDEX .) La couleur est représenté par un type entier color, mais tous les tampons d'indicateurs doivent avoir le typedouble. On ne peut pas recevoir les valeurs colorées et auxiliaires (INDICATOR_CALCULATIONS) des tampons à l'aide de la fonctionCopyBuffer().

Le nombre de tampons d'indicateur doit être indiqué par la directive du compilateur#property indicator_buffers le nombre_de tampons:

```
#property indicator_buffers 3  //  l'indicateur a 3 tampons
```

Le nombre maximum de tampons dans un indicateur est 512.

La conformité des tampons d'indicateur et des constructions graphiques

Chaque construction graphique est basé sur un ou plusieurs tampons d'indicateur. Ainsi pour un simple affichage des chandeliers japonais il est nécessaire quatre valeurs - les prix Open, High, Low et Close. En conséquence, pour afficher l'indicateur en forme de chandeliers japonais il faut déclarer 4 tampons d'indicateur, et 4 tableaux de type double pour eux. Par exemple:

```
//--- il y a quatre tampons d'indicateur dans l'indicateur
#property indicator_buffers 4
//--- il y a une construction graphique dans un indicateur
#property indicator_plots   1
//--- la construction graphique sous le numéro 1 s'affichera par les chandeliers japor
#property indicator_type1   DRAW_CANDLES
//--- Les chandeliers japonaises se dessineront par la couleur clrDodgerBlue
#property indicator_color1  clrDodgerBlue
//--- 4 tableaux pour les tampons d'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

Les constructions graphiques utilisent automatiquement les tampons d'indicateur conformément au numéro de la construction. Les numéros de la construction commencent par 1, les numéros des tampons commencent par le zéro. Si la première construction nécessite 4 tampons d'indicateur, les 4 premiers tampons d'indicateur seront utilisés pour le dessin. Ces quatre tampons doivent être liés par la fonction SetIndexBuffer()aux tableaux correspondants avec l'indexation juste.

```
//--- l'enchaînement des tableaux avec les tampons d'indicateur
   SetIndexBuffer(0,OBuffer,INDICATOR_DATA);  // le premier tampon correspond à l'inde
   SetIndexBuffer(1,HBuffer,INDICATOR_DATA);  // le deuxième tampon correspond à l'inc
   SetIndexBuffer(2,LBuffer,INDICATOR_DATA);  // le troisième tampon correspond à l'ir
   SetIndexBuffer(3,CBuffer,INDICATOR_DATA);  // le quatrième tampon correspond à l'ir
```

Pendant qu'on fait le dessin des chandeliers japonais l'indicateur utilisera notamment les premiers quatre tampons, parce que la construction "les chandeliers japonais" a été déclarée par le premier numéro.

Changeons un peu l'exemple, ajoutons la construction en forme de la ligne simple - DRAW_LINE. Supposons maintenant que la ligne aura le numéro 1 et les chandeliers japonais seront sous le numéro 2. Le nombre de tampons et le nombre de constructions a augmenté.

```
//--- dans l'indicateur il y a 5 tampons d'indicateur
#property indicator_buffers 5
//--- dans l'indicateur il y a 2 constructions graphiques
#property indicator_plots   2
//--- la construction graphique sous le numéro 1 s'affichera par la ligne
#property indicator_type1   DRAW_LINE
//--- la ligne sera dessinée par la couleur clrDodgerRed
#property indicator_color1  clrDodgerRed
//--- la construction graphique sous le numéro 2 s'affichera par les chandeliers japor
#property indicator_type2   DRAW_CANDLES
//--- Les chandeliers japonaises se dessineront par la couleur clrDodgerBlue
#property indicator_color2  clrDodgerBlue
//--- 5 tableaux pour les tampons d'indicateur
double LineBuffer[];
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

L'héritage des constructions a changé, maintenant par le premier il y a une ligne, après suivent les chandeliers japonais. C'est pourquoi l'ordre de l'héritage des tampons sera le même - d'abord nous déclarons le tampon pour la ligne sous l'index zéro, et puis quatre tampons pour l'affichage des chandeliers japonais.

```
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);  // le deuxième tampon correspond à l'
```

© 2000-2024, MetaQuotes Ltd.

```
//---l'enchaînement des tableaux avec les tampons d'indicateur pour les chandeliers ja
   SetIndexBuffer(1,OBuffer,INDICATOR_DATA);     // le deuxième tampon correspond à l
   SetIndexBuffer(2,HBuffer,INDICATOR_DATA);     // le troisième tampon correspond à l
   SetIndexBuffer(3,LBuffer,INDICATOR_DATA);     // le quatrième tampon correspond à l
   SetIndexBuffer(4,CBuffer,INDICATOR_DATA);     // le cinquième tampon correspond à l
```

On peut spécifier le nombre de tampons et de constructions graphiques seulement à l'aide des directives du compilateur, le changement dynamique de ces propriétés à l'aide des fonctions est impossible.

Les versions colorées des styles

Comme on peut voir dans le tableau, les styles se divisent sur deux groupes. Le premier groupe — ce sont les styles dont dans ses titres il n'y a pas le mot **COLOR**, appellerons-les les styles de base:

- DRAW_LINE
- DRAW_SECTION
- DRAW_HISTOGRAM
- DRAW_HISTOGRAM2
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_FILLING
- DRAW_BARS
- DRAW_CANDLES

Le deuxième groupe des styles contient dans le nom le mot **COLOR**, appellerons-les leurs versions colorées:

- DRAW_COLOR_LINE
- DRAW_COLOR_SECTION
- DRAW_COLOR_HISTOGRAM
- DRAW_COLOR_HISTOGRAM2
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG
- DRAW_COLOR_BARS
- DRAW_COLOR_CANDLES

Toutes les versions colorées des styles se distinguent des sujets de base qui permettent de spécifier la couleur pour chaque partie de la construction graphique. La partie minimale de la construction est le barre, c'est pourquoi on peut dire que les versions colorées permettent de spécifier la couleur de la construction sur chaque barre.

Pour spécifier la couleur de la construction sur chaque barre, aux versions colorées des styles a été ajouté le tampon supplémentaire spécial pour stocker l'index de la couleur. Ces index indiquent au numéro de la couleur dans un tableau spécial contenant l'ensemble prédéfini de couleurs. La taille du

© 2000-2024, MetaQuotes Ltd.

tableau des couleurs est 64. Cela signifie que chaque version colorée du style permet de colorer la construction graphique par 64 couleurs différentes.

On peut spécifier l'ensemble et le nombre de couleurs dans un tableau spécial des couleurs par la directive du compilateur #property indicator_color, où on indique toutes les couleurs nécessaires par la virgule. Par exemple, une telle inscription dans l'indicateur:

```
//--- spécifions 8 couleurs pour colorer les bougies (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
```

Ici est indiqué que pour la construction graphique le numéro 1 on spécifie 8 couleurs, qui seront placées au tableau spécial. Ensuite dans le programme nous spécifions pas la couleur elle-même, par laquelle s'affichera la construction graphique, mais seulement son index. Si nous voulons spécifierla couleur rouge pour la barre, pour cela il est nécessaire d'installer l'index de la couleur rouge du tableau dans un tampon coloré. La couleur rouge est spécifiée dans la directive par le première, l'index le numéro 0 lui correspond.

```
  //--- spécifions la couleur de la bougie clrRed
  col_buffer[buffer_index]=0;
```

L'ensemble des couleurs pour le coloriage n'est pas spécifié une fois pour toujours, on peut le changer dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Exemple:

```
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          color_array[i]);      //  une nouvelle couleur
```

Les propriétés de l'indicateur et des constructions graphiques

Pour les constructions graphiques on peut installer les propriétés à l'aidedirectives du compilateur, ainsi qu'à l'aide des fonctions appropriées. Plus en détail cela est décrit dans le paragrapheLien entre les propriétés de l'indicateur et les fonctions. Le changement dynamique des propriétés de l'indicateur à l'aide des fonctions permet de créer les indicateurs plus flexibles d'utilisateur.

Commencer à dessiner l'indicateur sur le graphique

En plusieurs cas sous les conditions de l'algorithme il est impossible de commencer le calcul des valeurs de l'indicateur tout de suite par la barre actuelle, il faut assurer le minimum des bars précédents accessibles dans l'histoire. Par exemple, plusieurs aspects de lissage sous-entendent qu'on prend le tableau des prix aux barres N précédentes, et sur la base de ces valeurs est calculée la valeur de l'indicateur pour une barre actuelle.

Dans tels cas il n'y a pas de possibilité de calculer les valeurs de l'indicateur sur les premières barres N, ou ces valeurs ne sont pas destinées à l'affichage sur le graphique et ils sont seulement auxiliaire pour le calcul des valeurs ultérieures. Pour refuser la visualisation de l'indicateur sur les premières barres N de l'histoire, il faut mettre à la propriété PLOT_DRAW_BEGIN la valeur N pour la construction correspondante graphique:

© 2000-2024, MetaQuotes Ltd.

```
//---l'enchaînement des tableaux avec les tampons d'indicateur pour les chandeliers ja
PlotIndexSetInteger(le numéro_de la construction_graphique,PLOT_DRAW_BEGIN,N);
```

Ici:

- le numéro_de la construction_graphique – la valeur du zéro jusqu'au indicator_plots-1 (le numérotage des constructions graphiques commence par le zéro).
- N – le nombre de premières barres dans l'histoire, sur lesquelles l'indicateur ne doit pas s'afficher sur le graphique.

© 2000-2024, MetaQuotes Ltd.

DRAW_NONE

Le style DRAW_NONE est conçu pour les cas quand il est nécessaire de calculer et afficher les valeurs du tampon dans "La fenêtre des données", mais l'affichage sur le graphique n'est pas nécessaire. Pour le réglage de l'exactitude de l'affichage utilisez l'expression IndicatorSetInteger(INDICATOR_DIGITS,le nombre_des signes)dans la fonction OnInit() :

```
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- établissons l'exactitude, avec laquelle la valeur sera affichée dans la Fenêtre
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
```

Le nombre de tampons nécessaires pour la construction DRAW_NONE — 1.

L'exemple de l'indicateur, qui affiche dans "la Fenêtre des données" le numéro de la barre, sur lequelle se trouve la souris. La numérotation correspond à la série temporelle, c'est-à-dire la barre inachevée actuel a l'index null, et la plus ancienne barre a le plus grand index.

Prêtez attention que malgré le fait qu'à la construction graphique №1 est spécifiée la couleur rouge de l'affichage, l'indicateur ne dessine rien sur le graphique.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_NONE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Invisible
#property indicator_label1  "Bar Index"
#property indicator_type1   DRAW_NONE
#property indicator_style1  STYLE_SOLID
#property indicator_color1  clrRed
#property indicator_width1  1
//--- indicator buffers
double         InvisibleBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le binding du tableau et du tampon d'indicateur
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- établissons l'exactitude, avec laquelle la valeur sera affichée dans "la Fenêtre
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static datetime lastbar=0;
//--- si c'est le premier calcul de l'indicateur
   if(prev_calculated==0)
     {
      //--- numéroterons les barres pour la première fois
      CalcValues(rates_total,close);
      //--- retiendrons le temps de l'ouverture de la barre actuelle dans lastbar
      lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
```

```
      }
    else
      {
       //---s'il y avait une nouvelle barre, le temps de son ouverture ne coïncide pas
       if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
         {
          //--- numéroterons les barres de nouveau
          CalcValues(rates_total,close);
          //---mettons à jour le temps de l'ouverture de la barre actuelle dans lastbar
          lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| numérote les barres comme dans la série temporelle
//+------------------------------------------------------------------+
void CalcValues(int total,double const  &array[])
  {
//--- spécifions l'indexation au tampon d'indicateur comme dans la série temporelle
   ArraySetAsSeries(InvisibleBuffer,true);
//--- remplirons à chaque barre son numéro
   for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
  }
```

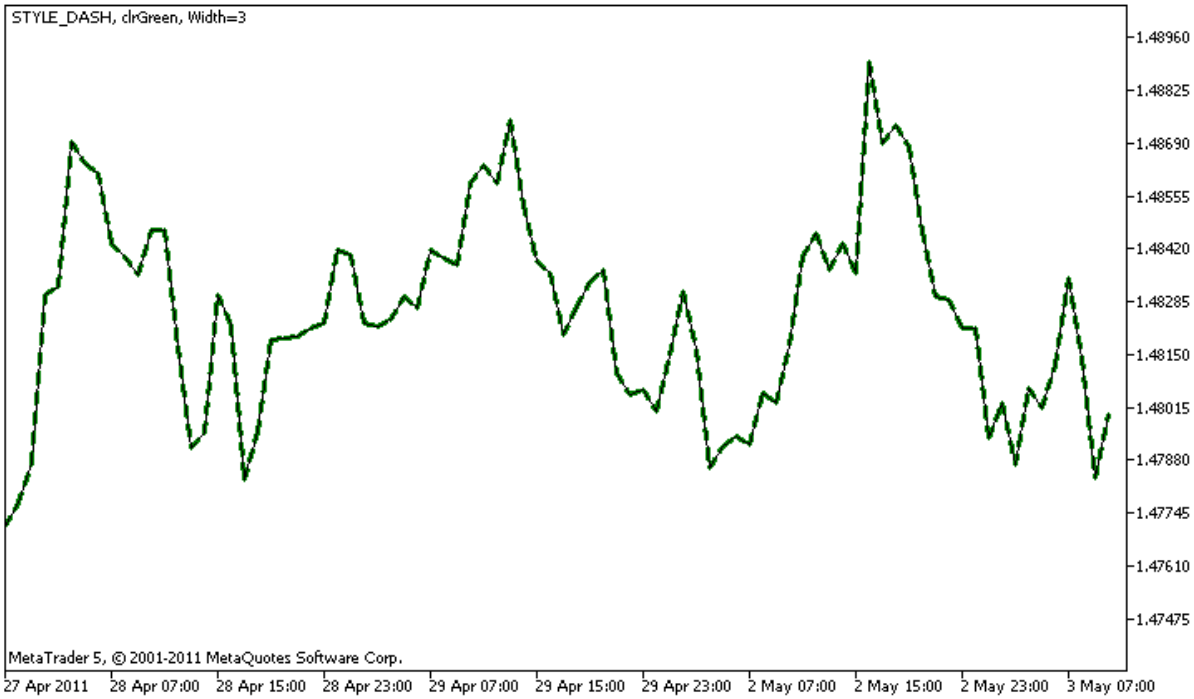
© 2000-2024, MetaQuotes Ltd.

DRAW_LINE

Le style DRAW_LINE dessine par la couleur spécifiée la ligne selon les valeurs du tampon d'indicateur. On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la ligne comme par les directives du compilateur, et dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

Le nombre de tampons nécessaires pour la construction DRAW_LINE — 1.

L'exemple de l'indicateur dessinant la ligne aux prix de la clôture des barres Close. La couleur, l'épaisseur et le style de la ligne changent chaques 5 ticks par hasard.



Prêtez attention que primordialement à la construction graphique**plot1** avec le style DRAW_LINE on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() ces trois propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_LINE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_LINE"
#property description "Dessine la ligne par la couleur spécifiée selon les prix Close"
#property description "La couleur, l'épaisseur et le style de la ligne change par hasa
```

```
#property description "dans chaques N ticks"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- les propriétés de la ligne sont spécifiées à l'aide des directives du compilateu
#property indicator_label1  "Line"      // le nom de la construction pour "la Fenêtre
#property indicator_type1   DRAW_LINE   // le type de la construction graphique - la l
#property indicator_color1  clrRed      // la couleur de la ligne
#property indicator_style1  STYLE_SOLID // le style des lignes
#property indicator_width1  1           // l'épaisseur de la ligne
//--- le paramètre input
input int      N=5;         // le nombre de ticks pour le changement
//--- le tampon d'indicateur pour la construction
double         LineBuffer[];
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le binding du tableau et du tampon d'indicateur
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- l'initialisation du générateur des nombres pseudo-accidentels
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
```

© 2000-2024, MetaQuotes Ltd.

```
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le bloc du calcul des valeurs de l'indicateur
   for(int i=0;i<rates_total;i++)
     {
      LineBuffer[i]=close[i];
     }

//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de la couleur de la ligne
//--- recevrons le nombre accidentel
   int number=MathRand();
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la ligne
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+", Width="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   size=ArraySize(styles);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm=EnumToString(styles[style_index])+", "+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_SECTION

Le style DRAW_SECTION dessine par la couleur spécifiée les segments selon les valeurs du tampon d'indicateur. On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la ligne comme pour le style DRAW_LINE - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

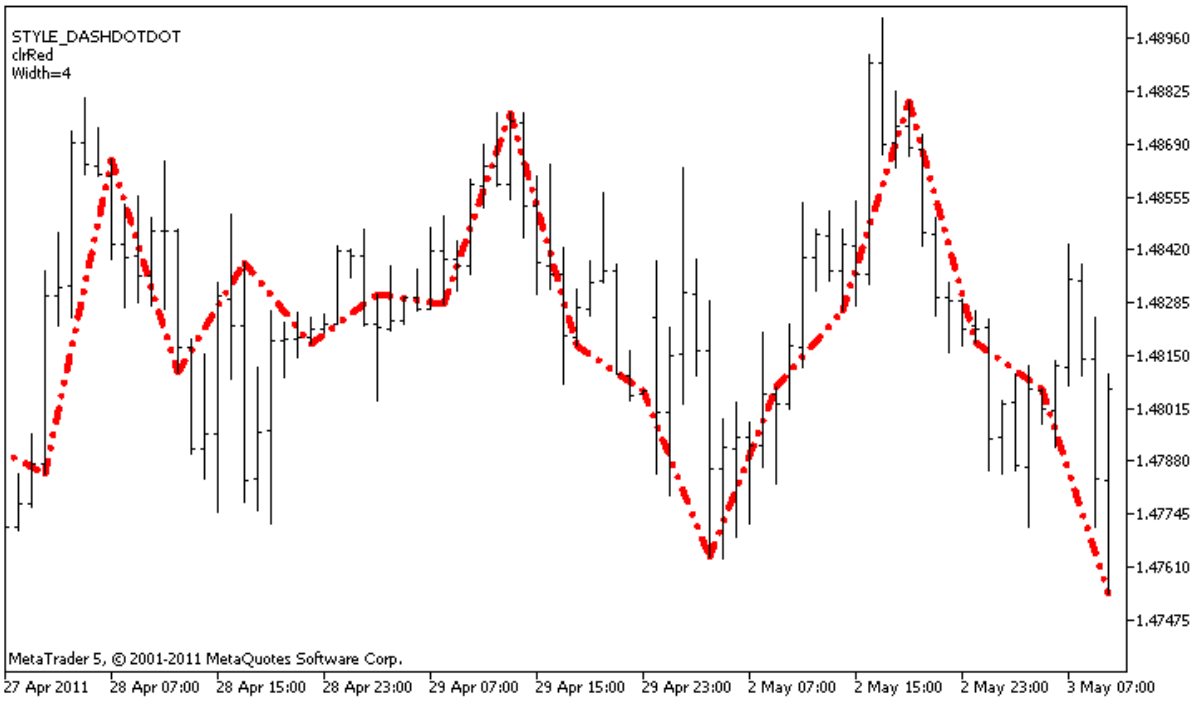
Les sections se dessinent d'une valeur non vide à l'autre valeur non vide du tampon d'indicateur, les valeurs vides sont manquées. Pour indiquer, quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriété PLOT_EMPTY_VALUE. Par exemple, si l'indicateur doit se dessiner par les segments selon les valeurs non nulles, il faut spécifier la valeur nulle comme vide:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(l'index_de la construction_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs tous les éléments du tampon d'indicateur, spécifiez aux éléments non-rendus la valeur vide.

Le nombre de tampons nécessaires pour la construction DRAW_SECTION — 1.

L'exemple de l'indicateur dessinant les segments entre les prix High et Low. La couleur, l'épaisseur et le style de **toutes** les sections se changent chaques**N** ticks par hasard.



Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_SECTION on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() ces trois propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                DRAW_SECTION.mq5 |
```

```
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_SECTION"
#property description "Dessine par les segments directs dans chaque "bars" barres"
#property description "La couleur, l'épaisseur et le style des sections se changent pa
#property description "dans chaques N ticks"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Section
#property indicator_label1  "Section"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- le paramètre input
input int      bars=5;           // la longueur des sections dans les barres
input int      N=5;              // le nombre de ticks pour le changement du style des
//--- le tampon d'indicateur pour la construction
double         SectionBuffer[];
//---la variable auxiliaire pour le calcul des fins des sections
int            divider;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le binding du tableau et du tampon d'indicateur
   SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---vérifions le paramètre de l'indicateur
   if(bars<=0)
     {
      PrintFormat("La valeur inadmissible du paramètre bar=%d",bars);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=2*bars;
//---+
```

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le numéro de la barre, par laquelle nous commencerons le calcul des valeurs de l
   int start=0;
//--- si l'indicateur a été déjà calculé, établissons start sur la barre précédente
   if(prev_calculated>0) start=prev_calculated-1;
//--- Tous les calculs des valeurs de l'indicateur se trouvent ici
   for(int i=start;i<rates_total;i++)
     {
      //--- recevrons le reste de la division du numéro de la barre sur 2*bars
      int rest=i%divider;
      //--- si le numéro de la barre se divise sans reste sur 2*bars
      if(rest==0)
        {
         //---fixerons la fin du segment sur le prix High de cette barre
         SectionBuffer[i]=high[i];
        }
      //--- si le reste de la division est égal aux barres,
      else
        {
         //---fixerons la fin du segment sur le prix High de cette barre
         if(rest==bars) SectionBuffer[i]=low[i];
         //--- si rien n'est pas convenu, manquons cette barre - mettons la valeur 0
```

```
          else SectionBuffer[i]=0;
         }
      }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| modifie l'apparence des sections dans l'indicateur                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de la couleur de la ligne
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la ligne
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- établissons l'épaisseur
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- établissons le style des lignes
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

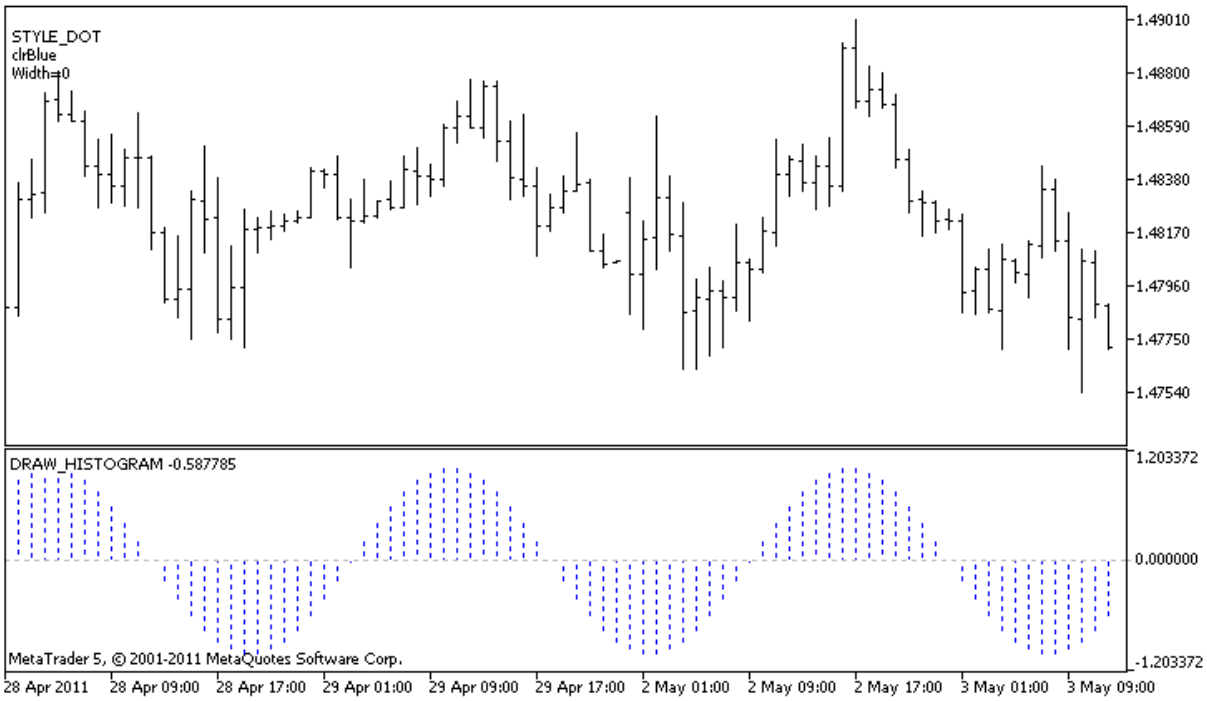
DRAW_HISTOGRAM

Le style DRAW_HISTOGRAM dessine par la couleur spécifiée l'histogramme par les colonnes du zéro jusqu'à la valeur indiquée. Les valeurs sont tirées du tampon de l'indicateur. On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la colonne comme pour le style DRAW_LINE - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'histogramme en fonction de la situation actuelle.

Puisque sur chaque barre se dessine la colonne du niveau nul, alors il est mieux utiliser DRAW_HISTOGRAM pour l'affichage dans une sous- fenêtre séparée du graphique. Le plus souvent ce type de la construction graphique est utilisé pour la création des indicateurs du type d'oscillateur, par exemple, Bears Power ou OsMA. Il suffit de spécifier la valeur nulle pour les valeurs vides non affichées.

Le nombre de tampons nécessaires pour la construction DRAW_HISTOGRAM — 1.

L'exemple de l'indicateur, dessinant la sinusoïde selon la fonctionMathSin()par la couleur spécifiée. La couleur, l'épaisseur et le style de**toutes** colonnes de l'histogramme se changent chaques **N** ticks par hasard. Le paramètre "bars" spécifie la période de la sinusoïde, c'est-à-dire dans le nombre spécifié de barres la sinusoïde se répétera selon la boucle.



Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_HISTOGRAM on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonctionOnCalculate() ces trois propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
```

```
//|                                                  https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_HISTOGRAM"
#property description "Dessine la sinusoïde par l'histogramme dans la fenêtre séparée"
#property description "La couleur, l'épaisseur des colonnes se changent par hasard"
#property description "dans chaques N ticks"
#property description "Le paramètre "bars" spécifie le nombre de barres dans la boucle

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres input
input int      bars=30;          // la période de la sinusoïde dans les barres
input int      N=5;              // le nombre de ticks pour le changement de l'histogr
//--- indicator buffers
double         HistogramBuffer[];
//--- le multiplicateur pour la réception de l'angle 2Pi dans les radians à la multipl
double    multiplier;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- calculerons le multiplicateur
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Spécifiez la valeur "bars"=%d plus que 1",bars);
      //--- une cessation anticipée du travail de l'indicateur
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
```

```
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- les calculs des valeurs de l'indicateur
   int start=0;
//--- si le calcul est déjà fait pendant le lancement précédent de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // établissons le début des calculs
//--- remplissons le tampon d'indicateur par les valeurs
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=sin(i*multiplier);
     }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| modifie l'apparence des lignes dans l'indicateur                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de la couleur de la ligne
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
```

© 2000-2024, MetaQuotes Ltd.

```
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la ligne
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- établissons l'épaisseur
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- établissons le style des lignes
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_HISTOGRAM2

Le style DRAW_HISTOGRAM2 dessine l'histogramme par la couleur spécifiée - les segments verticaux selon les valeurs de deux tampons d'indicateur. On peut spécifier l'épaisseur, la couleur et le style de l'affichage des segments comme pour le style DRAW_LINE - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'histogramme en fonction de la situation actuelle.

On peut utiliser le style DRAW_HISTOGRAM dans une sous- fenêtre séparée du graphique, ainsi que dans une fenêtre principale. On ne fait pas le dessin pour les valeurs vides, toutes les valeurs dans les tampons d'indicateur doivent être installés de manière explicite. L'initialisation des tampons par la valeur vide n'est pas produite.

Le nombre de tampons nécessaires pour la construction DRAW_HISTOGRAM2 — 2.

L'exemple de l'indicateur dessinant sur chaque barre le segment vertical de la couleur spécifiée et les épaisseurs entre les prix Open et Close. La couleur, l'épaisseur et le style de **toutes** colonnes de l'histogramme se changent chaques **N** ticks par hasard. Au lancement de l'indicateur dans la fonction OnInit()le numéro du jour de la semaine s'établit par l'image accidentelle, auquel l'histogramme ne sera pas dessiné - invisible_day. La valeur vide PLOT_EMPTY_VALUE=0: est spécifiée pour cela

```
//---établissons la valeur vide
   PlotIndexSetDouble(l'index_de la construction_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_HISTOGRAM2 on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() ces trois propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
```

```
//|                                                  DRAW_HISTOGRAM2.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                  https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_HISTOGRAM2"
#property description "Dessine le segment entre Open et Close sur chaque barre"
#property description "La couleur, l'épaisseur et le style se changent par hasard"
#property description "dans chaques N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot Histogram_2
#property indicator_label1  "Histogram_2"
#property indicator_type1   DRAW_HISTOGRAM2
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      N=5;              // le nombre de ticks pour le changement de l'histog
//--- indicator buffers
double         Histogram_2Buffer1[];
double         Histogram_2Buffer2[];
//--- le jour de la semaine, pour lequel l'indicateur n'est pas dessiné
int invisible_day;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//---établissons la valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- recevrons le nombre accidentel de 0 jusqu'à 5
   invisible_day=MathRand()%6;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- les calculs des valeurs de l'indicateur
   int start=0;
//--- pour la réception du jour de la semaine selon le temps de l'ouverture de chaque
   MqlDateTime dt;
//--- si le calcul est déjà fait pendant le lancement précédent de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // établissons le début des calculs
//--- remplissons le tampon d'indicateur par les valeurs
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      if(dt.day_of_week==invisible_day)
        {
         Histogram_2Buffer1[i]=0;
         Histogram_2Buffer2[i]=0;
        }
      else
        {
         Histogram_2Buffer1[i]=open[i];
         Histogram_2Buffer2[i]=close[i];
        }
     }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| modifie l'apparence des lignes dans l'indicateur                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de la couleur de la ligne
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la ligne
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- établissons l'épaisseur des lignes
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- établissons le style des lignes
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//---ajoutons l'information sur le jour, qui est manqué dans les calculs
   comm="\r\nLe jour non-dessiné - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+cor
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_ARROW

Le style DRAW_ARROW dessine les flèches sur le graphique par la couleur spécifiée (les symboles de l'ensemble Wingdings) selon la valeur du tampon d'indicateur. On peut spécifier l'épaisseur et la couleur des symboles de la meme façon que pour le style DRAW_LINE - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'indicateur en fonction de la situation actuelle.

Le code du symbole pour la sortie sur le graphique est spécifié à l'aide de la propriété PLOT_ARROW.

```
//--- - spécifions le code du symbole de l'ensemble Wingdings pour le dessin dans PLOT
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

Par défaut la valeur PLOT_ARROW=159 (le cercle).

Chaque flèche représente en réalité le symbole, qui a la hauteur et le point du rattachement, et peut fermer par lui-même une certaine importante information sur le graphique (par exemple, le prix de la clôture de la barre). C'est pourquoi on peut spécifier en supplément un décalage vertical en pixels, qui ne dépend pas de l'échelle du graphique. Les flèches seront visuellement déplacées selon la verticale sur le nombre indiqué de pixels, bien que les valeurs de l'indicateur restent les mêmes:

```
//--- spécifions le décalage des flèches selon le vertical en pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

La valeur négative PLOT_ARROW_SHIFT signifie le décalage des flèches en haut, la valeur positive déplace les flèches en bas.

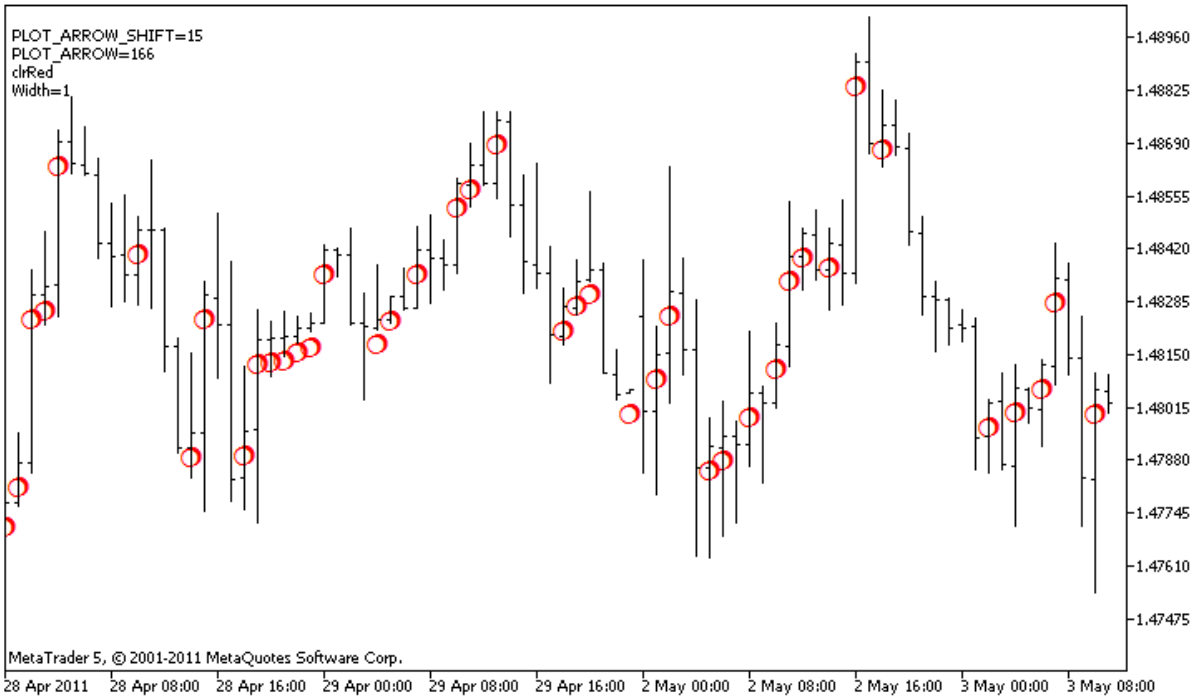
On peut utiliser le style DRAW_ARROW dans une sous- fenêtre séparée du graphique, ainsi que dans une fenêtre principale. Les valeurs vides ne se dessinent pas et ne s'affichent pas dans "la Fenêtre des données", toutes les valeurs dans les tampons d'indicateur doivent être installés de manière explicite. L'initialisation des tampons par la valeur vide n'est pas produite.

```
//---établissons la valeur vide
   PlotIndexSetDouble(l'index_de la construction_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

Le nombre de tampons nécessaires pour la construction DRAW_ARROW — 1.

L'exemple de l'indicateur dessinant les flèches sur chaque barre, qui a le prix de la clôture Close plus grand que le prix de la clôture de la barre précédente La couleur, l'épaisseur et le code du symbole de**toutes**les flèches se changent chaques **N** ticks par hasard.

© 2000-2024, MetaQuotes Ltd.



Dans l'exemple les propriétés la couleur et la taille pour la construction graphique **plot1** avec le style DRAW_ARROW sont spécifiés primordialement à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate()les propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_ARROW.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_ARROW"
#property description "Dessine les flèches, spécifiées par les symboles Unicode, sur l
#property description "La couleur, la taille, le décalage et le code du symbole de la
#property description "dans chaques N ticks"
#property description "Le paramètre code spécifie la valeur de base: le code = 159 (ce

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Arrows
#property indicator_label1  "Arrows"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrGreen
#property indicator_width1  1
//--- les paramètres input
```

```
input int      N=5;         // le nombre de ticks pour le changement
input ushort   code=159;    // le code du symbole pour le dessin dans DRAW_ARROW
//--- le tampon d'indicateur pour la construction
double         ArrowsBuffer[];
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- spécifions le code du symbole pour le dessin dans PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- spécifions le décalage des flèches selon le vertical en pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- définissons 0 comme une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement de la couleur, de la taille, de la confus
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le bloc du calcul des valeurs de l'indicateur
```

```
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- la boucle du calcul
   for(int i=1;i<rates_total;i++)
     {
      //--- si le prix courant Close est plus grand que le précédent, mettons la flèch
      if(close[i]>close[i-1])
         ArrowsBuffer[i]=close[i];
      //--- dans le cas contraire spécifions la valeur nulle
      else
         ArrowsBuffer[i]=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| modifie l'apparence des symboles dans l'indicateur                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de l'indicateur
   string comm="";
//--- le bloc du changement de la couleur de la flèche
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de la taille des flèches
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // on spécifie la taille de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons la taille des flèches
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du code de la flèche (PLOT_ARROW)
   number=MathRand();
//---recevrons le reste de la division entière pour le calcul du nouveau code de la fl
   int code_add=number%20;
//---définissons un nouveau code du symbole comme la somme code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- inscrirons le code du symbole PLOT_ARROW
```

© 2000-2024, MetaQuotes Ltd.

```
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- le bloc du changement du décalage des flèches selon la verticale en pixels
   number=MathRand();
//--- recevrons le décalage comme le reste de la division entière
   int shift=20-number%41;
//--- définissons un nouveau décalage de-20 jusqu'à 20
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- inscrirons le décalage PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_ZIGZAG

Le style DRAW_ZIGZAG dessine par la couleur spécifiée les segments selon les valeurs de deux tampons d'indicateur. Ce style est très semblable auDRAW_SECTION, mais à la différence du dernier permet de dessiner les segments verticaux dans la limite d'une barre, si on a spécifié les valeurs pour les deux tampons d'indicateur pour cette barre. Les segments se dessinent de la valeur dans un premier tampon jusqu'à la valeur dans un deuxième tampon d'indicateur. Aucun des tampons ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la ligne comme pour le style DRAW_SECTION - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

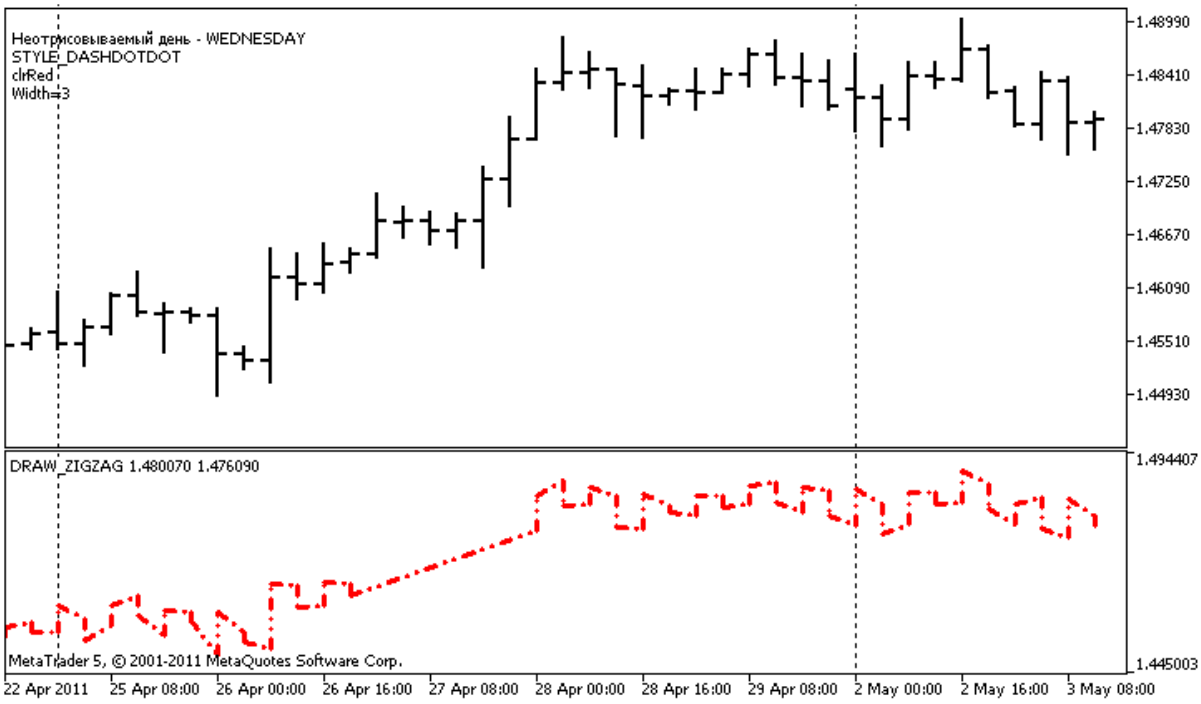
Les sections se dessinent d'une valeur non vide à l'autre valeur non vide du tampon d'indicateur. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriétéPLOT_EMPTY_VALUE:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(l'index_de la construction_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

Le nombre de tampons nécessaires pour la construction DRAW_ZIGZAG — 2.

L'exemple de l'indicateur dessinant la scie de long selon les prix High et Low. La couleur, l'épaisseur et le style des lignes du zigzag se changent par hasard chaque **N** ticks.



Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_ZIGZAG on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() ces trois propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les

© 2000-2024, MetaQuotes Ltd.

paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_ZIGZAG.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_ZIGZAG"
#property description "Dessine par les segments directs \"la scie \", en manquant les
#property description "Le jour des manques est choisi par hasard au lancement de l'inc
#property description "La couleur, l'épaisseur et le style des segments se changent"
#property description " par hasard par chaque N ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ZigZag
#property indicator_label1  "ZigZag"
#property indicator_type1   DRAW_ZIGZAG
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres input
input int      N=5;              // le nombre de ticks pour le changement
//--- indicator buffers
double         ZigZagBuffer1[];
double         ZigZagBuffer2[];
//--- le jour de la semaine, pour lequel l'indicateur n'est pas dessiné
int invisible_day;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- l'enchaînement des tableaux et des tampons d'indicateur
   SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- recevrons le nombre accidentel de 0 jusqu'à 6, pour ce jour l'indicateur n'est p
   invisible_day=MathRand()%6;
//---la valeur 0 (la valeur vide) ne participera pas au rendu
```

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- la structure du temps sera nécessaire pour la réception du jour de la semaine de
   MqlDateTime dt;

//--- la position du commencement des calculs
   int start=0;
//---si l'indicateur a été calculé sur le tick précédent, nous commençons le calcul pa
   if(prev_calculated!=0) start=prev_calculated-1;
//--- la boucle des calculs
   for(int i=start;i<rates_total;i++)
     {
      //--- inscrirons le temps de l'ouverture de la barre dans la structure
      TimeToStruct(time[i],dt);
      //--- si le jour de la semaine cette barre est égal à invisible_day
      if(dt.day_of_week==invisible_day)
        {
         //--- inscrirons les valeurs vides aux tampons pour cette barre
         ZigZagBuffer1[i]=0;
```

```
          ZigZagBuffer2[i]=0;
        }
      //--- si le jour de la semaine est convenable, remplissons les tampons
      else
        {
         //--- si le numéro de la barre est pair
         if(i%2==0)
           {
            //---   écrivons au 1-er tampon High, et au 2-ème Low
            ZigZagBuffer1[i]=high[i];
            ZigZagBuffer2[i]=low[i];
           }
         //--- le numéro de la barre est impair
         else
           {
            //--- remplissons la barre dans l'ordre inverse
            ZigZagBuffer1[i]=low[i];
            ZigZagBuffer2[i]=high[i];
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| modifie l'apparence des segments dans le zigzag                  |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de l'indicateur Z
   string comm="";
//--- le bloc du changement de la couleur du zigzag
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la ligne
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
```

© 2000-2024, MetaQuotes Ltd.

```
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//---ajoutons l'information sur le jour, qui est manqué dans les calculs
   comm="\r\nLe jour non-dessiné - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+cor
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_FILLING

Le style DRAW_FILLING dessine le domaine coloré entre les valeurs de deux tampons d'indicateur. En fait ce style dessine deux lignes et peint l'espace entre eux par une de deux couleurs spécifiées. Il est destiné à la création des indicateurs dessinant les canaux. Aucun des tampons ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

On peut spécifier deux couleurs de remplissage

- la première couleur pour ces domaines, où les valeurs dans le premier tampon d'indicateur sont plus grandes que les valeurs dans le deuxième tampon d'indicateur;
- la deuxième couleur pour ces domaines, où les valeurs dans le deuxième tampon d'indicateur sont plus grandes que les valeurs dans le premier tampon d'indicateur.

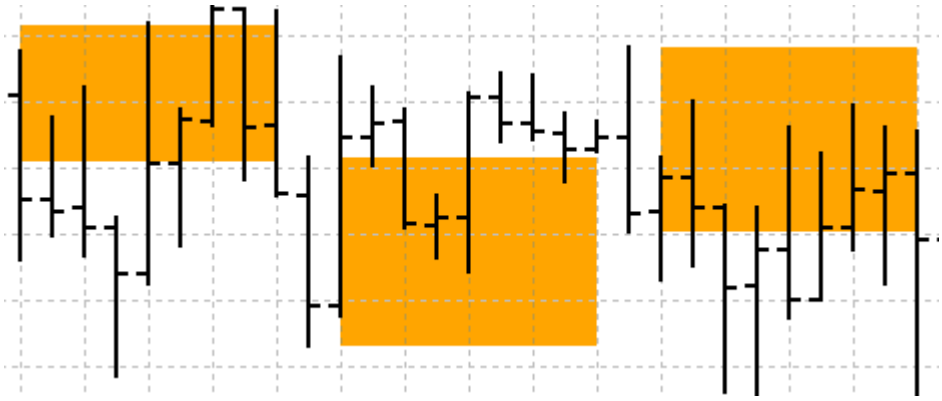
Vous pouvez spécifier la couleur de remplissagepar les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de "vivifier" les indicateurs pour qu'ils changent l'aspect en fonction de la situation actuelle.

L'indicateur est calculé pour toutes les barres, pour lesquelles les valeurs des deux tampons d'indicateur ne sont pas égales à 0 et ne sont pas égales à la valeur nulle. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriétéPLOT_EMPTY_VALUE:

```
   #define INDICATOR_EMPTY_VALUE -1.0
   ...
//--- la valeur INDICATOR_EMPTY_VALUE (une valeur vide) ne participera pas dans le cal
   PlotIndexSetDouble(L'indice_de la construction_DRAW_FILLING,PLOT_EMPTY_VALUE,INDICA
```

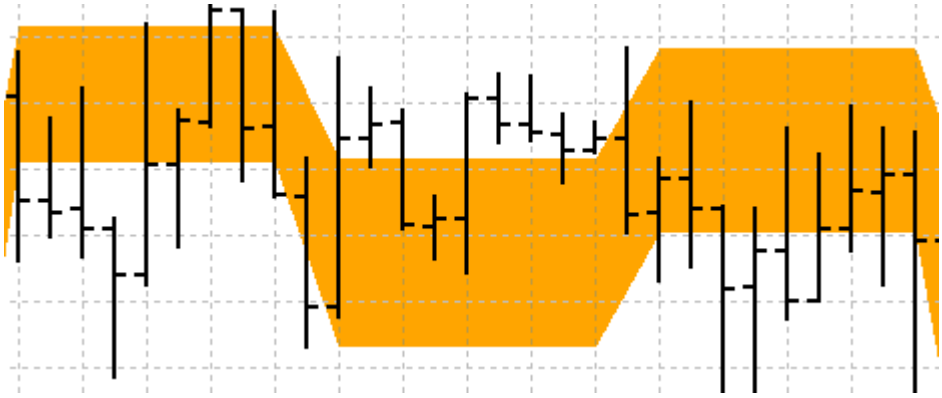
La rendu sur les barres, qui ne participent pas au calcul de l'indicateur, dépendra des valeurs dans les tampons d'indicateur:

- Les barres pour lesquelles les valeurs des deux tampons d'indicateurs sont égales à 0, ne participent pas à la rendu de l'indicateur. C'est-à-dire le domaine avec les valeurs nulles ne sera pas coloré.



- Les barres pour lesquelles les valeurs des deux tampons d'indicateurs sont égales à la valeur vide, ne participent pas à la rendu de l'indicateur. Le domaine avec les valeurs vides sera coloré de telle manière pour joindre les domaines avec les valeurs significatives.

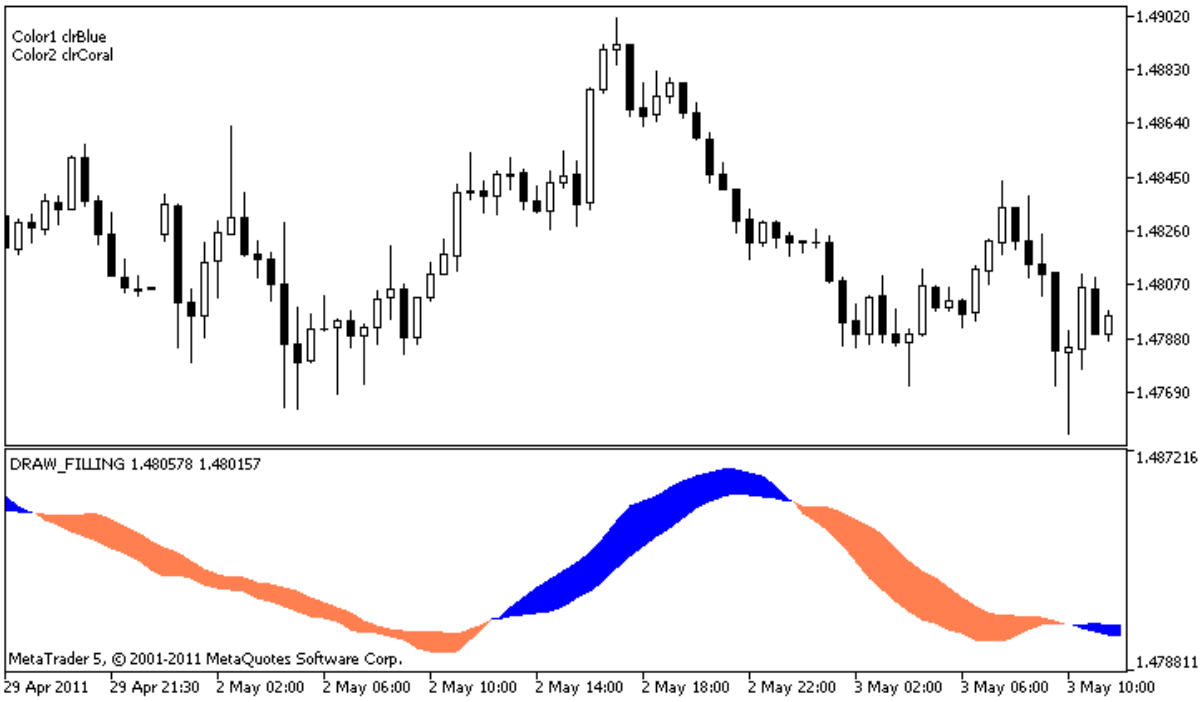
© 2000-2024, MetaQuotes Ltd.



Il est important de marquer que si "une valeur vide" est égale au zéro, les barres qui ne participant pas dans le calcul de l'indicateur seront aussi colorées.

Le nombre de tampons nécessaires pour la construction DRAW_FILLING — 2.

L'exemple de l'indicateur dessinant dans la fenêtre séparée le canal entre deux glissants moyens avec différentes périodes de la prise de moyenne. Le changement de la couleur à l'intersection des moyens montre visuellement le remplacement des tendances montante et descendante. Les couleurs se changent chaques**N** ticks par hasard. Le paramètre N est porté auxparamètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans les propriétés de l'indicateur).



Prêtez attention que primordialement à la construction graphique**plot1** avec le style DRAW_FILLING on spécifie deux couleurs à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() les nouvelles couleurs sont spécifiées par hasard.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_FILLING.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_FILLING"
#property description "Dessine dans la fenêtre séparée le canal entre deux moyens"
#property description "La couleur du remplissage du canal se change par hasard"
#property description "dans chaques N ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot Intersection
#property indicator_label1  "Intersection"
#property indicator_type1   DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1  1
//--- les paramètres input
input int      Fast=13;          // la période de la glissante moyenne rapide
input int      Slow=21;          // la période de la glissante moyenne lente
input int      shift=1;          // le décalage des moyennes au futur (positif)
input int      N=5;              // le nombre de ticks pour le changement
//--- les tampons d'indicateur
double         IntersectionBuffer1[];
double         IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrC
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,IntersectionBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,IntersectionBuffer2,INDICATOR_DATA);
//---
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//---
   fast_handle=iMA(_Symbol,_Period,Fast,0,MODE_SMA,PRICE_CLOSE);
   slow_handle=iMA(_Symbol,_Period,Slow,0,MODE_SMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//---faisons le premier calcul de l'indicateur ou les données sont changées et on dema
   if(prev_calculated==0)
     {
      //--- copions toutes les valeurs des indicateurs dans les tampons correspondants
      int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
     }
   else // remplissons économiquement seulement les données, qui sont mises à jour
     {
      //--- recevrons la différence dans les barres entre le lancement actuel et précé
      int to_copy=rates_total-prev_calculated;
      //--- s'il n'y a pas de différence, en tout cas copions une seule valeur - sur l
      if(to_copy==0) to_copy=1;
      //--- copions to_copy des valeurs à la fin de tampons d'indicateur
      int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| change les couleurs du remplissage du canal                      |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
```

```
   string comm="";
//--- le bloc du changement de la couleur de la ligne
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);

//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index1=number%size;
//--- définissons la première couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
//--- inscrirons la première couleur
   comm=comm+"\r\nColor1 "+(string)colors[color_index1];

//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   number=MathRand(); // recevrons le nombre accidentel
   int color_index2=number%size;
//--- définissons la deuxième couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
//--- inscrirons la deuxième couleur
   comm=comm+"\r\nColor2 "+(string)colors[color_index2];
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_BARS

Le style DRAW_BARS dessine les barres selon les valeurs de quatre tampons d'indicateur, qui contiennent les prix Open, High, Low et Close. Il est destiné à la création des indicateurs personnels en forme des barres, y compris dans une sous- fenêtre séparée du graphique et selon d'autres instruments financiers.

On peut spécifier la couleur des barres par les directives du compilateur ou dynamiquement à l'aide de la fonctionPlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de "vivifier" les indicateurs pour qu'ils changent l'aspect en fonction de la situation actuelle.

L'indicateur est dessiné seulement pour ces barres, pour lesquelles on a spécifié les valeurs non vides de **tous** les quatre tampons d'indicateur. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriété PLOT_EMPTY_VALUE:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(индекс_построения_DRAW_BARS,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

Le nombre de tampons nécessaires pour la construction DRAW_BARS — 4. Tous les tampons pour la construction doivent aller successivement un après l'autre en ordre donné: Open, High, Low et Close. Aucun des tampons ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

L'exemple de l'indicateur dessinant les barres selon l'instrument financier indiqué dans la fenêtre séparée. La couleur des barres se change chaques **N** ticks par hasard. Le paramètre N est porté auxparamètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans les propriétés de l'indicateur).

© 2000-2024, MetaQuotes Ltd.

Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_BARS on spécifie la couleur à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() la couleur est spécifiée par hasard . par hasard de la liste d'avance préparée.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_BARS.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_BARS"
#property description "Dessine dans la fenêtre séparée les barres selon le symbole cho
#property description "La couleur et l'épaisseur et le symbole des barres se changent
#property description "par chaque N ticks"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "Bars"
#property indicator_type1   DRAW_BARS
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres input
input int      N=5;              // le nombre de ticks pour le changement de l'aspect
input int      bars=500;         // combien de barres montrer
input bool     messages=false;   // l'affichage des messages dans le journal "Experts"
//--- les tampons d'indicateur
double         BarsBuffer1[];
double         BarsBuffer2[];
double         BarsBuffer3[];
double         BarsBuffer4[];
//--- le nom du symbole
string symbol;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- s'il y a trop peu de barres - terminons le travail avant terme
   if(bars<50)
     {
      Comment("Spécifiez un plus grand nombre de barres! Le travail de l'indicateur es
```

```
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
//---le nom du symbole, selon lequel les barres se dessinent
   symbol=_Symbol;
//--- établissons l'affichage du symbole
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- choisirons un nouveau symbole de la fenêtre "Aperçu du marché"
      symbol=GetRandomSymbolName();
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();

      int tries=0;
      //--- faisons 5 tentatives de remplir les tampons par les prix du symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
         //--- le compteur des appels de la fonction CopyFromSymbolToBuffers()
         tries++;
        }
```

```
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur par les prix                |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total)
  {
//--- copions les prix Open, High, Low et Close au tableau rates[]
   MqlRates rates[];
//--- le compteur des tentatives
   int attempts=0;
//--- combien on a été copié
   int copied=0;
//--- faisons 25 tentatives de recevoir la série temporelle selon le symbole nécessai
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si on n'a pas réussi à copier le nombre suffisant des barres
   if(copied!=bars)
     {
      //--- formons la chaîne du message
      string comm=StringFormat("On a réussi à recevoir seulement %d barres de %d barre
                               name,
                               copied,
                               bars
                               );
      //--- déduisons le message au commentaire sur une principale fenêtre du graphiqu
      Comment(comm);
      //--- déduisons les messages
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- établissons l'affichage du symbole
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+name+")");
     }
//--- initialisons les tampons par les valeurs vides
   ArrayInitialize(BarsBuffer1,0.0);
   ArrayInitialize(BarsBuffer2,0.0);
   ArrayInitialize(BarsBuffer3,0.0);
```

© 2000-2024, MetaQuotes Ltd.

```
   ArrayInitialize(BarsBuffer4,0.0);
//--- copions les prix aux tampons
   for(int i=0;i<copied;i++)
     {
      //--- calculons l'index correspondant pour les tampons
      int buffer_index=total-copied+i;
      //--- inscrivons les prix aux tampons
      BarsBuffer1[buffer_index]=rates[i].open;
      BarsBuffer2[buffer_index]=rates[i].high;
      BarsBuffer3[buffer_index]=rates[i].low;
      BarsBuffer4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Rend accidentellement le symbole du Market Watch                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- le nombre de symboles montrés dans la fenêtre "Aperçu du marché"
   int symbols=SymbolsTotal(true);
//--- la position du symbole dans la liste - le nombre accidentel de 0 jusqu'aux symbo
   int number=MathRand()%symbols;
//--- rendons le nom du symbole selon la position indiquée
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence des barres                                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés des barres
   string comm="";
//--- le bloc du changement de la couleur des barres
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la barre
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
```

© 2000-2024, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- inscrirons le nom du symbole
   comm="\r\n"+symbol+comm;

//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_CANDLES

Le style DRAW_CANDLES dessine les chandeliers japonais selon les valeurs de quatre tampons d'indicateur, qui contiennent les prix Open, High, Low et Close. Il est destiné à la création des indicateurs personnels en forme des bougies, y compris dans une sous- fenêtre séparée du graphique et selon d'autres financiers financiers.

On peut spécifier la couleur des barres par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de "vivifier" les indicateurs pour qu'ils changent l'aspect en fonction de la situation actuelle.

L'indicateur est dessiné seulement pour ces barres, pour lesquelles on a spécifié les valeurs non vides de **tous** les quatre tampons d'indicateur. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriété PLOT_EMPTY_VALUE:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(индекс_построения_DRAW_CANDLES,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

Le nombre de tampons nécessaires pour la construction DRAW_CANDLES — 4. Tous les tampons pour la construction doivent aller successivement un après l'autre en ordre donné: Open, High, Low et Close. Aucun des tampons ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

You can set up to three colors for the DRAW_CANDLES style affecting the candle look. If only one color is set, it is applied to all candles on a chart.

```
//--- identical candles with a single color applied to them
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- only one color is specified, therefore all candles are of the same color
#property indicator_color1  clrGreen
```

If two comma-separated colors are specified, the first one is applied to candle outlines, while the second one is applied to the body.

```
//--- different colors for candles and wicks
#property indicator_label1  "Two color candles"
#property indicator_type1   DRAW_CANDLES
//--- green is applied to wicks and outlines, while white is applied to the body
#property indicator_color1  clrGreen,clrWhite
```

Specify three comma-separated colors so that rising and falling candles are displayed differently. In that case, the first color is applied to the candle outlines, while the second and third ones - to bullish and bearish candles.

```
//--- different colors for candles and wicks
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- wicks and outlines are green, bullish candle body is white, while bearish candle
```

© 2000-2024, MetaQuotes Ltd.

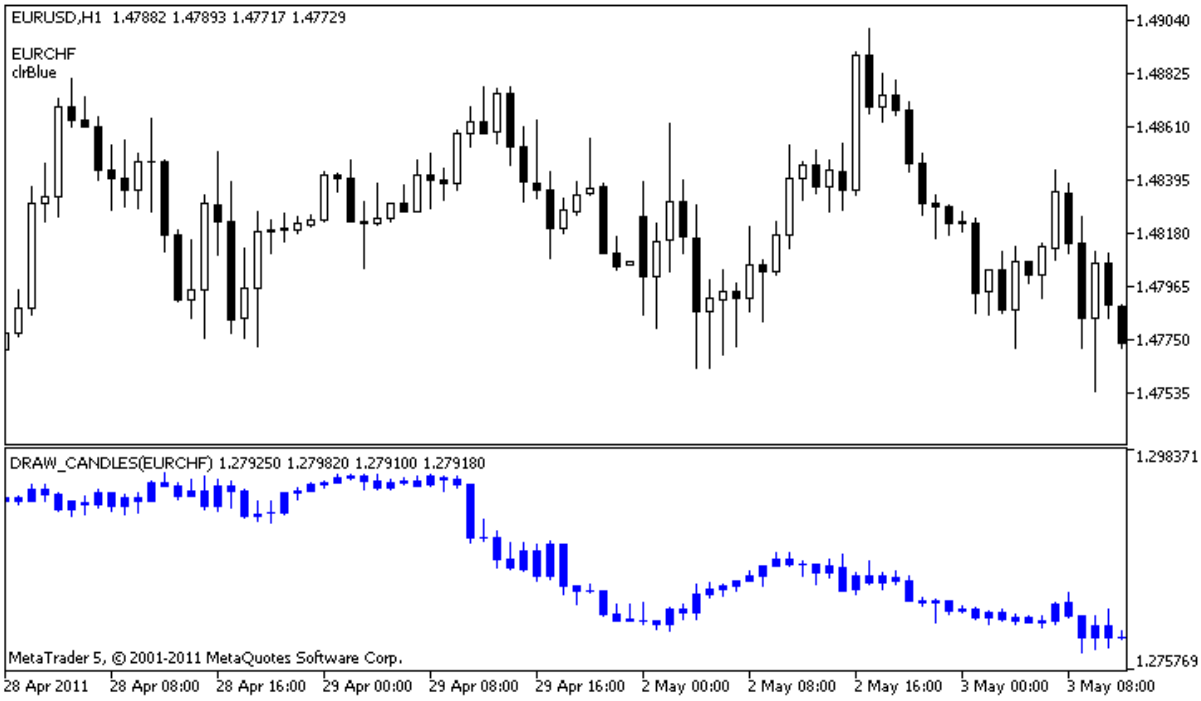
```
#property indicator_color1  clrGreen,clrWhite,clrRed
```

Thus, the DRAW_CANDLES style allows you to create custom candle coloring options. Besides, all colors can be changed dynamically during the indicator operation using the PlotIndexSetInteger function (composition_index_DRAW_CANDLES, PLOT_LINE_COLOR, modifier_index, color), where modifier_index may have the following values:

- 0 - colors of outlines and wicks
- 1- bullish candle body color
- 2 - bearish candle body color

```
//--- set the color of outlines and wicks
PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrBlue);
//--- set the bullish body color
PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrGreen);
//--- set the bearish body color
PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrRed);
```

L'exemple de l'indicateur dessinant les chandeliers japonaises selon l'instrument financier indiqué dans la fenêtre séparée. La couleur des chandeliers se change chaques**N** ticks par hasard. Le paramètre N est porté auxparamètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans les propriétés de l'indicateur).



Prêtez attention que primordialement à la construction graphique **plot1**on spécifie la couleur à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() une nouvelle couleur est spécifiée par hasard de la liste d'avance préparée.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_CANDLES.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
```

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_CANDLES."
#property description "Dessine dans la fenêtre séparée par les couleurs différentes le
#property description " "
#property description "La couleur et l'épaisseur et le symbole des bougies se changent
#property description "par hasard par chaque N ticks."

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "DRAW_CANDLES1"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- les paramètres input
input int      N=5;              // le nombre de ticks pour le changement de l'aspect
input int      bars=500;         // combien de barres montrer
input bool     messages=false;   // l'affichage des messages dans le journal "Experts"
//--- les tampons d'indicateur
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- le nom du symbole
string symbol;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- s'il y a trop peu de barres - terminons le travail avant terme
   if(bars<50)
     {
      Comment("Spécifiez un plus grand nombre de barres! Le travail de l'indicateur es
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
   SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
```

```
//--- une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---le nom du symbole, selon lequel les barres se dessinent
   symbol=_Symbol;
//--- établissons l'affichage du symbole
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES("+symbol+")");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- choisirons un nouveau symbole de la fenêtre "Aperçu du marché"
      symbol=GetRandomSymbolName();
      //--- remplaçons l'aspect
      ChangeLineAppearance();
      //--- choisirons un nouveau symbole de la fenêtre "Aperçu du marché"
      int tries=0;
      //--- faisons 5 tentatives de remplir les tampons plot1 par les prix du symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
        {
         //--- le compteur des appels de la fonction CopyFromSymbolToBuffers()
         tries++;
        }
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Remplit une bougie indiquée                                      |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[]
                             )
  {
//--- copions les prix Open, High, Low et Close au tableau rates[]
   MqlRates rates[];
//--- le compteur des tentatives
   int attempts=0;
//--- combien on a été copié
   int copied=0;
//--- faisons 25 tentatives de recevoir la série temporelle selon le symbole nécessai
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attem
     }
//--- si on n'a pas réussi à copier le nombre suffisant des barres
   if(copied!=bars)
     {
      //--- formons la chaîne du message
      string comm=StringFormat("On a réussi à recevoir seulement %d barres de %d barre
                               name,
                               copied,
                               bars
                               );
      //--- déduisons le message au commentaire sur une principale fenêtre du graphiq
      Comment(comm);
      //--- déduisons les messages
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- établissons l'affichage du symbole
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
     }
//--- initialisons les tampons par les valeurs vides
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
```

© 2000-2024, MetaQuotes Ltd.

```
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- copions les prix aux tampons sur chaque tick
   for(int i=0;i<copied;i++)
     {
      //--- calculons l'index correspondant pour les tampons
      int buffer_index=total-copied+i;
      //--- inscrivons les prix aux tampons
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Rend accidentellement le symbole du Market Watch                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- le nombre de symboles montrés dans la fenêtre "Aperçu du marché"
   int symbols=SymbolsTotal(true);
//--- la position du symbole dans la liste - le nombre accidentel de 0 jusqu'aux symbo
   int number=MathRand()%symbols;
//--- rendons le nom du symbole selon la position indiquée
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence des barres                                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés des barres
   string comm="";
//--- le bloc du changement de la couleur des barres
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur
   comm=comm+"\r\n"+(string)colors[color_index];
//--- inscrirons le nom du symbole
   comm="\r\n"+symbol+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_LINE

Le style DRAW_COLOR_LINE c'est la variante colorée du style DRAW_LINE, il dessine aussi la ligne selon les valeur du tampon d'indicateur. Mais ce style, comme tous les styles de couleur qui ont **COLOR**dans le titre, a un tampon supplémentaire spécial d'indicateur pour stocker l'index (le numéro) de la couleur du tableau spécial spécifié des couleurs. Ainsi, on peut spécifier la couleur de chaque endroit, si on indique les index de la couleur, par lesquels la ligne doit être dessinée sur une barre spécifiée.

On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la ligne comme par les directives du compilateur, et dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

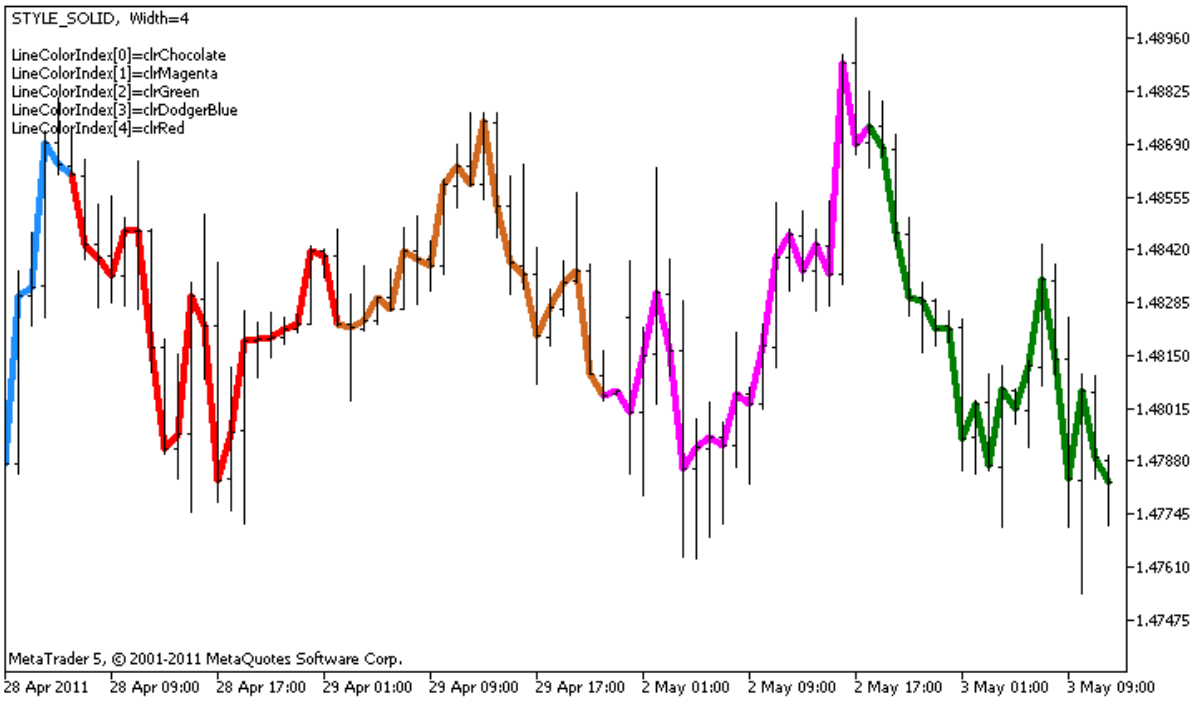
Le nombre de tampons nécessaires pour la construction DRAW_COLOR_LINE — 2:

- un tampon pour stocker la valeur de l'indicateur selon lequel se dessine la ligne;
- un tampon pour stocker l'index de la couleur, par lequel la ligne est dessinée sur chaque barre.

On peut spécifier les couleurs par la directive du compilateur #property indicator_color1 par la virgule. Le nombre de couleurs ne peut pas dépasser 64.

```
//--- spécifions 5 couleurs pour colorer chaque barre (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (on peut
```

L'exemple de l'indicateur dessinant la ligne aux prix de la clôture des barres Close. L'épaisseur et le style de la ligne se changent par hasard chaques N=5 ticks.



Les couleurs, par lesquelles les fragments de la ligne sont dessinés, se changent aussi par hasard dans la fonction d'utilisateur ChangeColors().

```
//+------------------------------------------------------------------+
```

```
//| Change la couleur des sections de la ligne                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
```

Dans l'exemple on montre la particularité des versions "de couleur" des indicateurs - pour changer la couleur du fragment de la ligne il ne faut pas changer la valeur dans le tampon ColorLineColors [] (qui contient les index des couleurs). Il suffit de spécifier les nouvelles couleurs dans un tableau spécial. Cela vous permet de changer rapidement la couleur pour toute la construction graphique, en changeant seulement un petit tableau des couleurs à l'aide de la fonctionPlotIndexSetInteger().

Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_COLOR_LINE on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() ces trois propriétés sont spécifiées par hasard.

Les paramètres N et Length (la longeur des fragments de couleur dans les barres) sont portés aux paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_LINE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "L'indicateur pour la démonstration DRAW_COLOR_LINE"
#property description "Dessine la ligne par les morceaux colorés selon 20 barres aux p
#property description "La couleur, l'épaisseur et le style des fragments de la ligne s
#property description "par chaque N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
//--- spécifions 5 couleurs pour colorer chaque barre (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (on peut
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres input
input int      N=5;           // le nombre de ticks pour le changement
input int      Length=20;     // la longueur de chaque fragment de la couleur en barre
int            line_colors=5; // le nombre de couleurs spécifiées est égale à 5 - rega
//--- le tampon pour le dessin
double         ColorLineBuffer[];
//--- le tableau pour le stockage de la couleur du dessin de la ligne sur chaque barre
double         ColorLineColors[];

//--- le tableau pour le stockage des couleurs contient 7 éléments
color colors[]={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGolde
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le binding du tableau et du tampon d'indicateur
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- l'initialisation du générateur des nombres pseudo-accidentels
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseu
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles les fragments de la ligne se dessin
      ChangeColors(colors,5);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le bloc du calcul des valeurs de l'indicateur
   for(int i=0;i<rates_total;i++)
     {
      //--- inscrirons la valeur de l'indicateur au tampon
      ColorLineBuffer[i]=close[i];
      //--- maintenant spécifions l'index de la couleur pour cette barre par hasard
      int color_index=i%(5*Length);
      color_index=color_index/Length;
      //--- pour cette barre la ligne sera dessinée par la couleur, qui se trouve sous
      ColorLineColors[i]=color_index;
     }

//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Change la couleur des sections de la ligne                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
```

```
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+" Width="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   int size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm=EnumToString(styles[style_index])+", "+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_SECTION

Le style DRAW_COLOR_SECTION c'est une variante colorée du style DRAW_SECTION, mais contrairement à ce dernier, permet de dessiner chaque section par sa propre couleur. Le style DRAW_COLOR_SECTION, comme tous les styles de couleur ayant**COLOR**dans le titre, contient le tampon supplémentaire spécial d'indicateur pour stocker l'index (le numéro) de la couleur du tableau spécial spécifié des couleurs. Ainsi, on peut spécifier la couleur de chaque section, si on indique les index de la couleur pour la barre, sur laquelle tombe la fin de la section.

On peut spécifier l'épaisseur, la couleur et le style des segments comme pour le style DRAW_SECTION - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

Les sections se dessinent d'une valeur non vide à l'autre valeur non vide du tampon d'indicateur, les valeurs vides sont manquées. Pour indiquer, quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriété PLOT_EMPTY_VALUE. Par exemple, si l'indicateur doit se dessiner par les segments selon les valeurs non nulles, il faut spécifier la valeur nulle comme vide:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(индекс_построения_DRAW_COLOR_SECTION,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs tous les éléments du tampon d'indicateur, spécifiez aux éléments non-rendus la valeur vide.

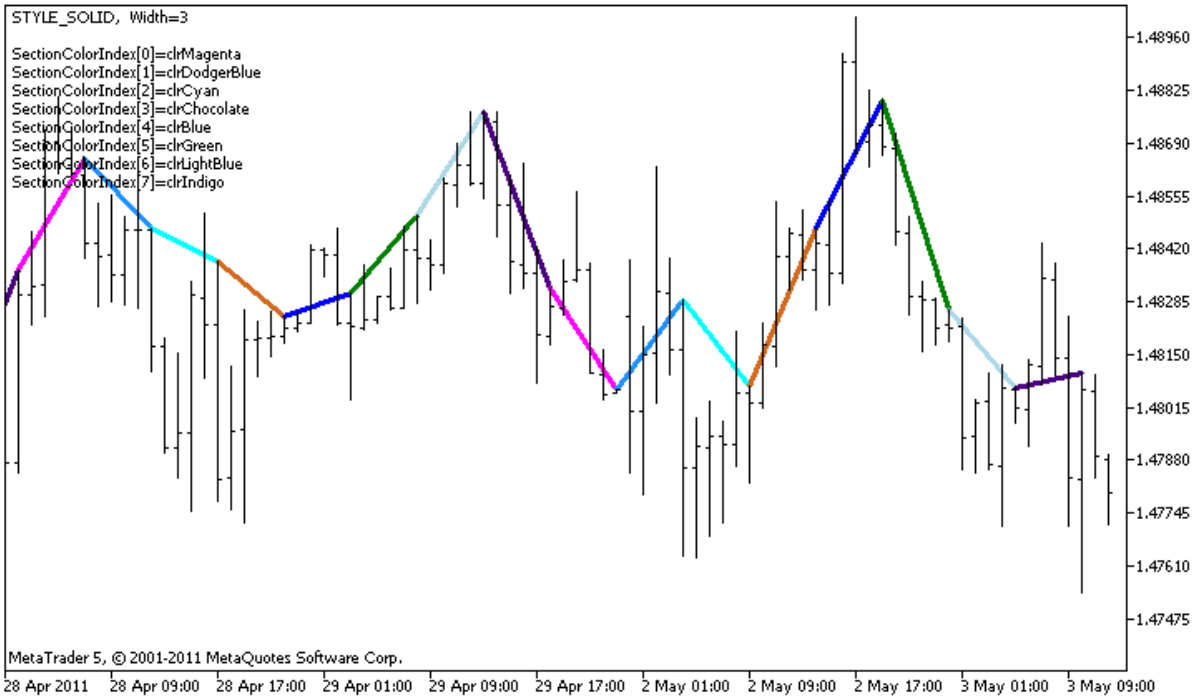
Le nombre de tampons nécessaires pour la construction DRAW_COLOR_SECTION — 2:

- un tampon pour stocker la valeur de l'indicateur selon lequel se dessine la ligne;
- un tampon pour stocker l'index de la couleur, par lequel la section est dessinée (il faut spécifier seulement pour les valeurs non vides).

On peut spécifier les couleurs par la directive du compilateur #property indicator_color1 par la virgule. Le nombre de couleurs ne peut pas dépasser 64.

L'exemple de l'indicateur dessinant les sections multicolores d'une longueur de 5 barres selon les prix High. La couleur, l'épaisseur et le style des sections se changent par hasard chaque**N** ticks.

© 2000-2024, MetaQuotes Ltd.



Prêtez attention que primordialement à la construction graphique**plot1** avec le style DRAW_COLOR_SECTION on spécifie 8 couleurs à l'aide de la directive du compilateur #property. Puis dans la fonction OnCalculate() les couleurs sont spécifiées par hasard du tableau colors[].

Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_SECTION.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_SECTION"
#property description "Dessine par les segments colorés de la longueur au nombre spéci
#property description "La couleur, l'épaisseur et le style des sections se changent pa
#property description "dans chaques N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorSection
#property indicator_label1  "ColorSection"
#property indicator_type1   DRAW_COLOR_SECTION
//--- spécifions 8 couleurs pour colorer les sections (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,c
#property indicator_style1  STYLE_SOLID
```

```
#property indicator_width1  1
//--- les paramètres input
input int      N=5;                      // le nombre de ticks pour le changement
input int      bars_in_section=5;        // la longueur des sections en barres
//---la variable auxiliaire pour le calcul des fins des sections
int            divider;
int            color_sections;
//--- le tampon pour le dessin
double         ColorSectionBuffer[];
//--- le tableau pour le stockage de la couleur du dessin de la ligne sur chaque barre
double         ColorSectionColors[];
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- le nombre de couleurs pour colorier les sections
   color_sections=8;   //  regarder le commentaire pour la propriété #property indicat
//---vérifions le paramètre de l'indicateur
   if(bars_in_section<=0)
     {
      PrintFormat("La valeur inadmissible de la longueur la section=%d",bars_in_sectio
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=color_sections*bars_in_section;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles les sections sont dessinées
      ChangeColors(colors,color_sections);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le numéro de la barre, par laquelle nous commencerons le calcul des valeurs de l
   int start=0;
//--- si l'indicateur a été déjà calculé, établissons start sur la barre précédente
   if(prev_calculated>0) start=prev_calculated-1;
//--- Tous les calculs des valeurs de l'indicateur se trouvent ici
   for(int i=start;i<rates_total;i++)
     {
      //--- si le numéro de la barre se divise sans reste en longueur_de la section, c
      if(i%bars_in_section==0)
        {
         //---fixerons la fin du segment sur le prix High de cette barre
         ColorSectionBuffer[i]=high[i];
         //--- le reste de la division du numéro de la barre sur longueur_de la sectio
         int rest=i%divider;
         //recevrons le numéro de la couleur =  de 0 jusqu'au nombre_des couleurs-1
         int color_indext=rest/bars_in_section;
         ColorSectionColors[i]=color_indext;
        }
      //--- si le reste de la division est égal aux barres,
      else
        {
         //--- si rien n'est pas convenu, manquons cette barre - mettons la valeur 0
         ColorSectionBuffer[i]=0;
        }
     }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Change la couleur des sections de la ligne                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("SectionColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStr
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence de la ligne affichée dans l'indicateur        |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+" Width="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   int size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
```

```
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm=EnumToString(styles[style_index])+", "+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM

Le style DRAW_COLOR_HISTOGRAM dessine l'histogramme par les colonnes du zéro jusqu'à la valeur indiquée. Les valeurs sont tirées du tampon de l'indicateur. Chaque colonne peut avoir sa propre couleur de l'ensemble des couleurs d'avance prédéterminé.

n peut spécifier l'épaisseur, la couleur et le style du histogramme comme pour le style DRAW_HISTOGRAM – par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'histogramme en fonction de la situation actuelle.

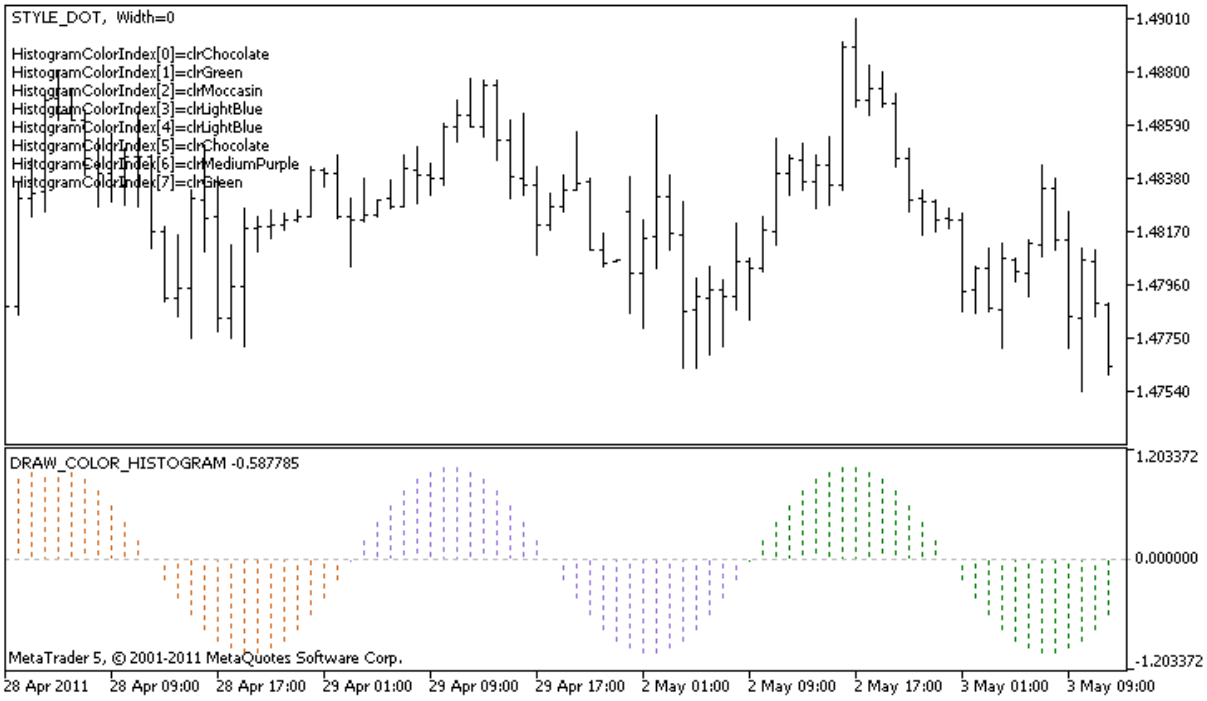
Puisque sur chaque barre se dessine la colonne du niveau nul, alors il est mieux d'utiliser DRAW_COLOR_HISTOGRAM pour l'affichage dans une sous- fenêtre séparée du graphique. Le plus souvent ce type de la construction graphique est utilisé pour la création des indicateurs du type d'oscillateur, par exemple, Awesome Oscillator ou Market Facilitation Index. Il suffit de spécifier la valeur nulle pour les valeurs vides non affichées.

Le nombre de tampons nécessaires pour la construction DRAW_COLOR_HISTOGRAM — 2:

- un tampon pour stocker de la valeur non nulle du segment vertical sur chaque barre, la deuxième fin du segment se trouve toujours sur la ligne zéro de l'indicateur;
- un tampon pour stocker l'index de la couleur, par lequel la section est dessinée (il faut spécifier seulement pour les valeurs non vides).

On peut spécifier les couleurs par la directive du compilateur #property indicator_color1 par la virgule. Le nombre de couleurs ne peut pas dépasser 64.

L'exemple de l'indicateur, dessinant la sinusoïde selon la fonctionMathSin()par la couleur spécifiée. La couleur, l'épaisseur et le style de**toutes** colonnes de l'histogramme se changent chaques **N** ticks par hasard. Le paramètre "bars" spécifie la période de la sinusoïde, c'est-à-dire dans le nombre spécifié de barres la sinusoïde se répétera selon la boucle.



© 2000-2024, MetaQuotes Ltd.

Pretez attention que primordialement pour la construction graphique **plot1** avec le style DRAW_COLOR_HISTOGRAM on spécifie 5 couleurs à l'aide de la directive du compilateur#property indicator_color1, et puis dans la fonction OnCalculate() les couleurs sont choisies par hasard parmi 14 couleurs stockées dans le tableau colors[]. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_HISTOGRAM"
#property description "Dessine la sinusoïde par l'histogramme dans la fenêtre séparée"
#property description "La couleur, l'épaisseur des colonnes se changent par hasard"
#property description "dans chaques N ticks"
#property description "Le paramètre "bars" spécifie le nombre de barres dans la boucle

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- les paramètres input
input int      bars=30;          // la période de la sinusoïde dans les barres
input int      N=5;              // le nombre de ticks pour le changement de l'histogr
//--- plot Color_Histogram
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- spécifions 8 couleurs pour colorer les sections (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrMe
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- le tampon des valeurs
double         Color_HistogramBuffer[];
//--- le tampon des index de la couleur
double         Color_HistogramColors[];
//--- le multiplicateur pour la réception de l'angle 2Pi dans les radians à la multipl
double         multiplier;
int            color_sections;
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- le tableau pour le stockage des styles du dessin de la ligne
```

```
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
//---- le nombre de couleurs pour colorier la sinusoïde
   color_sections=8;   //  regarder le commentaire pour la propriété #property indicat
//--- calculerons le multiplicateur
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Spécifiez la valeur "bars"=%d plus que 1",bars);
      //--- une cessation anticipée du travail de l'indicateur
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles l'histogramme est dessiné
      ChangeColors(colors,color_sections);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- les calculs des valeurs de l'indicateur
   int start=0;
//--- si le calcul est déjà fait pendant le lancement précédent de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // établissons le début des calculs
//--- remplissons le tampon d'indicateur par les valeurs
   for(int i=start;i<rates_total;i++)
     {
      //--- la valeur
      Color_HistogramBuffer[i]=sin(i*multiplier);
      //--- la couleur
      int color_index=i%(bars*color_sections);
      color_index/=bars;
      Color_HistogramColors[i]=color_index;
     }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Change la couleur des sections de la ligne                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+" Width="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   int size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm=EnumToString(styles[style_index])+", "+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM2

Le style DRAW_COLOR_HISTOGRAM2 dessine l'histogramme par la couleur spécifiée – les segments verticaux selon les valeurs de deux tampons d'indicateur. Mais à la différence de DRAW_HISTOGRAM2 monochrome on peut spécifier la couleur propre de l'ensemble prédéterminé dans ce style à chaque colonne de l'histogramme. Les valeurs des fins des segments sont tirées du tampon d'indicateur.

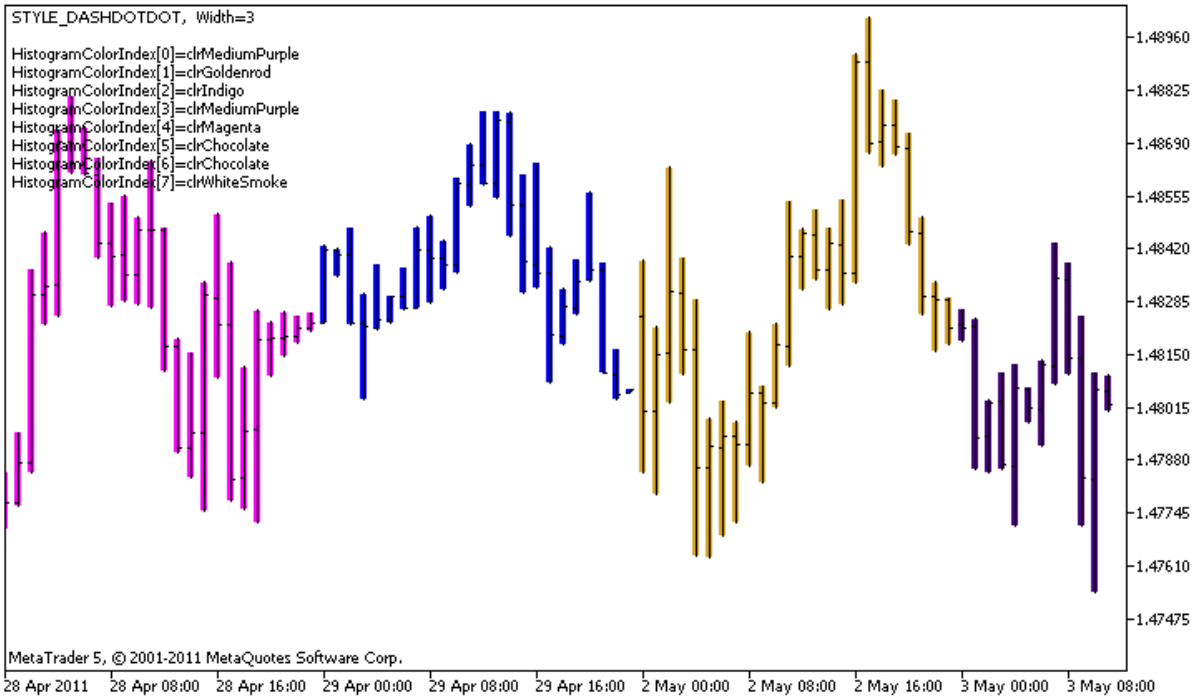
On peut spécifier l'épaisseur, le style et la couleur de l'histogramme comme pour le style DRAW_HISTOGRAM2 – par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'histogramme en fonction de la situation actuelle.

On peut utiliser le style DRAW_COLOR_HISTOGRAM2 dans une sous- fenêtre séparée du graphique, ainsi que dans une fenêtre principale. On ne fait pas le dessin pour les valeurs vides, toutes les valeurs dans les tampons d'indicateur doivent être installés de manière explicite. L'initialisation des tampons par la valeur vide n'est pas produite.

Le nombre de tampons nécessaires pour la construction DRAW_COLOR_HISTOGRAM2 — 3:

- deux tampons pour stocker l'extrémité supérieure et inférieure du segment vertical sur chaque barre;
- un tampon pour stocker l'index de la couleur, par lequel le segment est dessiné (il faut spécifier seulement pour les valeurs non vides).

L'exemple de l'indicateur, qui dessine l'histogramme entre les prix High et Low par la couleur spécifiée. Les lignes de l'histogramme se dessinent par sa couleur pour chaque jour de la semaine. La couleur de chaque jour, l'épaisseur et le style de l'histogramme se changent chaques**N** ticks par hasard.



Prêtez attention que primordialement à la construction graphique **plot1**avec le style DRAW_COLOR_HISTOGRAM2 on spécifie 5 couleurs à l'aide de la directive du compilateur #property indicator_color1, et puis dans la fonction OnCalculate()les couleurs sont choisies par hasard parmi 14 couleurs stockées dans le tableau colors[].

© 2000-2024, MetaQuotes Ltd.

Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM2.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_HISTOGRAM2"
#property description "Dessine le segment entre Open et Close sur chaque barre"
#property description "La couleur, l'épaisseur et le style se changent par hasard"
#property description "dans chaques N ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plot ColorHistogram_2
#property indicator_label1  "ColorHistogram_2"
#property indicator_type1   DRAW_COLOR_HISTOGRAM2
//--- spécifions 5 couleurs pour colorer l'histogramme selon les jours de la semaine
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- le paramètre input
input int      N=5;              // le nombre de ticks pour le changement de l'histogr
int            color_sections;
//--- les tampons des valeurs
double         ColorHistogram_2Buffer1[];
double         ColorHistogram_2Buffer2[];
//--- le tampon des index de la couleur
double         ColorHistogram_2Colors[];
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
```

```
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
//---établissons la valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- le nombre de couleurs pour colorier la sinusoïde
   color_sections=8;   //  regarder le commentaire à la propriété #property indicator_
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles l'histogramme est dessiné
      ChangeColors(colors,color_sections);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- les calculs des valeurs de l'indicateur
   int start=0;
//--- pour la réception du jour de la semaine selon le temps de l'ouverture de chaque
   MqlDateTime dt;
//--- si le calcul est déjà fait pendant le lancement précédent de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // établissons le début des calculs
//--- remplissons le tampon d'indicateur par les valeurs
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- la valeur
      ColorHistogram_2Buffer1[i]=high[i];
      ColorHistogram_2Buffer2[i]=low[i];
      //---spécifions l'index de la couleur par jour de la semaine
      int day=dt.day_of_week;
      ColorHistogram_2Colors[i]=day;
     }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Change la couleur des sections de la ligne                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
```

```
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+" Width="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   int size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm=EnumToString(styles[style_index])+", "+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_ARROW

Le style DRAW_COLOR_ARROW dessine les flèches sur le graphique par la couleur (les symboles de l'ensemble Wingdings) selon la valeur du tampon d'indicateur. A la différence du style DRAW_ARROW, on y peut spécifier la couleur pour chaque symbole de l'ensemble des couleurs prédéterminé, spécifiées par la propriété indicator_color1.

On peut spécifier l'épaisseur et la couleur des symboles de la meme façon que pour le style DRAW_ARROW - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'histogramme en fonction de la situation actuelle.

Le code du symbole pour la sortie sur le graphique est spécifié à l'aide de la propriété PLOT_ARROW.

```
//--- - spécifions le code du symbole de l'ensemble Wingdings pour le dessin dans PLOT
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

Par défaut la valeur PLOT_ARROW=159 (le cercle).

Chaque flèche représente en réalité le symbole, qui a la hauteur et le point du rattachement, et peut fermer par lui-même une certaine importante information sur le graphique (par exemple, le prix de la clôture de la barre). C'est pourquoi on peut spécifier en supplément un décalage vertical en pixels, qui ne dépend pas de l'échelle du graphique. Les flèches seront visuellement déplacées selon la verticale sur le nombre indiqué de pixels, bien que les valeurs de l'indicateur restent les mêmes:

```
//--- spécifions le décalage des flèches selon le vertical en pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

La valeur négative PLOT_ARROW_SHIFT signifie le décalage des flèches en haut, la valeur positive déplace les flèches en bas.

On peut utiliser le style DRAW_COLOR_ARROW dans une sous- fenêtre séparée du graphique, ainsi que dans une fenêtre principale.. Les valeurs vides ne se dessinent pas et ne s'affichent pas dans "la Fenêtre des données", toutes les valeurs dans les tampons d'indicateur doivent être installés de manière explicite. L'initialisation des tampons par la valeur vide n'est pas produite.

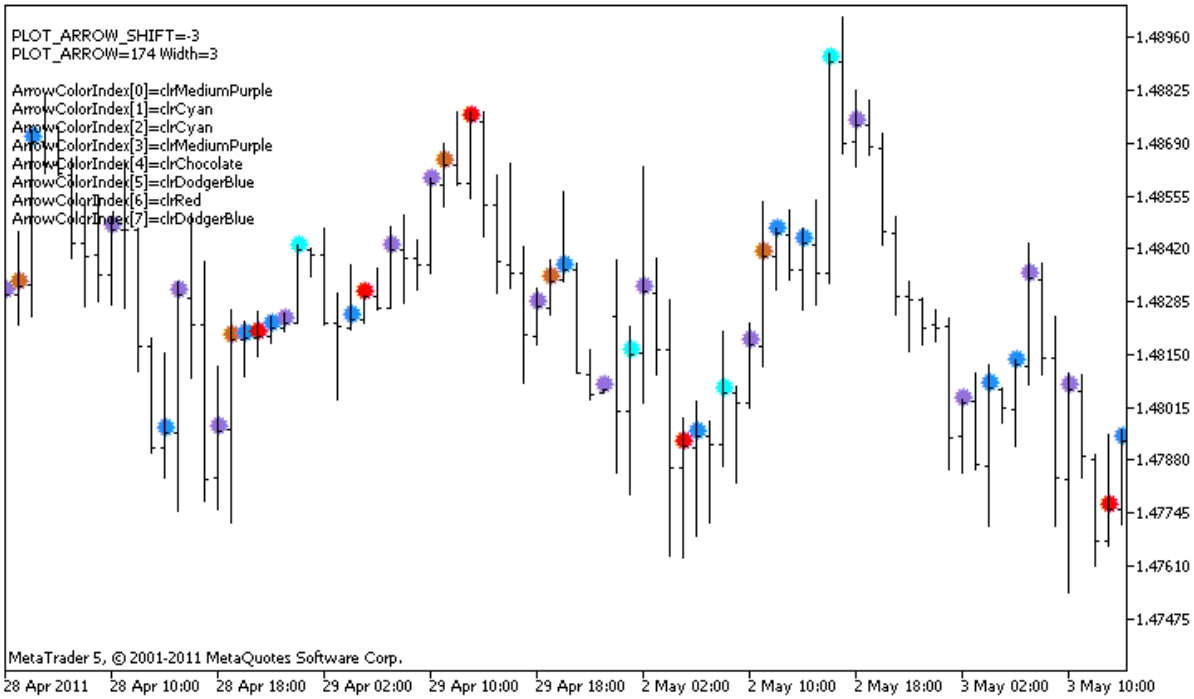
```
//---établissons la valeur vide
   PlotIndexSetDouble(l'index_de la construction_DRAW_COLOR_ARROW,PLOT_EMPTY_VALUE,0);
```

Le nombre de tampons nécessaires pour la construction DRAW_COLOR_ARROW — 2:

- un tampon pour stocker la valeur du prix selon lequel se dessine le symbole (en plus le décalage en pixels, spécifié par la propriété PLOT_ARROW_SHIFT);
- un tampon pour stocker l'index de la couleur, par lequel se dessine la flèche (il faut spécifier seulement pour les valeurs non vides).

L'exemple de l'indicateur dessinant les flèches sur chaque barre, qui a le prix de la clôture Close plus grand que le prix de la clôture de la barre précédente. L'épaisseur, le décalage et le code du symbole de**toutes** les flèches se changent chaques **N** ticks par hasard. La couleur du symbole dépend du numéro de la barre, sur lequelle il est dessiné.

© 2000-2024, MetaQuotes Ltd.



Dans l'exemple les propriétés la couleur et la taille pour la construction graphique **plot1** avec le style DRAW_COLOR_ARROW sont spécifiés primordialement à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate()les propriétés sont spécifiées par hasard.. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

Pretez attention que primordialement on spécifie 8 couleurs à l'aide de la directive du compilateur#property, et puis dans la fonctionOnCalculate() la couleur est choisie par hasard parmi 14 couleurs stockées dans le tableau colors[].

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_ARROW.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_ARROW"
#property description "Dessine sur le graphique par les différentes couleurs les flèch
#property description "La couleur, la taille, le décalage et le code du symbole de la
#property description " par hasard par chaque N ticks"
#property description "Le paramètre code spécifie la valeur de base: le code = 159 (ce

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorArrow
#property indicator_label1  "ColorArrow"
```

```
#property indicator_type1   DRAW_COLOR_ARROW
//--- spécifions 8 couleurs pour colorer l'histogramme selon les jours de la semaine
#property indicator_color1  clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagent
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- les paramètres input
input int      N=5;         // le nombre de ticks pour le changement
input ushort   code=159;    // le code du symbole pour le dessin dans DRAW_ARROW
int            color_sections;
//--- le tampon d'indicateur pour la construction
double         ColorArrowBuffer[];
//--- le tableau pour le stockage des index de la couleur
double         ColorArrowColors[];
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
//--- spécifions le code du symbole pour le dessin dans PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- spécifions le décalage des flèches selon le vertical en pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- définissons 0 comme une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- le nombre de couleurs pour colorier la sinusoïde
   color_sections=8;   //  regarder le commentaire pour la propriété #property indicat
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000-2024, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement de la couleur, de la taille, de la confus
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés des flèches
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles l'histogramme est dessiné
      ChangeColors(colors,color_sections);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le bloc du calcul des valeurs de l'indicateur
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- la boucle du calcul
   for(int i=1;i<rates_total;i++)
     {
      //--- si le prix courant Close est plus grand que le précédent, mettons la flèch
      if(close[i]>close[i-1])
         ColorArrowBuffer[i]=close[i];
      //--- dans le cas contraire spécifions la valeur nulle
      else
         ColorArrowBuffer[i]=0;
      //--- la couleur de la flèche
      int index=i%color_sections;
      ColorArrowColors[i]=index;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Change la couleur des sections de la ligne                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- recevrons le nombre accidentel
       int number=MathRand();
       //--- recevrons l'index dans le tableau col[] comme le reste de la division ent
       int i=number%size;
       //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLO
       PlotIndexSetInteger(0,                    //  le numéro du style graphique
                           PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                           plot_color_ind,       //  l'index de la couleur, où nous ins
                           cols[i]);             //   une nouvelle couleur
       //--- inscrirons les couleurs
       comm=comm+StringFormat("ArrowColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
       ChartSetString(0,CHART_COMMENT,comm);
      }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+" Width="+IntegerToString(width);

//--- le bloc du changement du code de la flèche (PLOT_ARROW)
   number=MathRand();
//---recevrons le reste de la division entière pour le calcul du nouveau code de la f
   int code_add=number%20;
//---définissons un nouveau code du symbole comme la somme code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- inscrirons le code du symbole PLOT_ARROW
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- le bloc du changement du décalage des flèches selon la verticale en pixels
   number=MathRand();
//--- recevrons le décalage comme le reste de la division entière
   int shift=20-number%41;
//--- définissons un nouveau décalage
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- inscrirons le décalage PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_ZIGZAG

Le style DRAW_COLOR_ZIGZAG dessine les segments des couleurs différentes selon les valeurs de deux tampons d'indicateur. Ce style est la version colorée du style DRAW_ZIGZAG, c'est-à-dire permet de spécifier à chaque segment une propre couleur de l'ensemble des couleurs d'avance prédéterminé. Les segments se dessinent de la valeur dans un premier tampon jusqu'à la valeur dans un deuxième tampon d'indicateur. Aucun des tampons ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la ligne comme pour le style DRAW_ZIGZAG - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

Les sections se dessinent d'une valeur non vide à l'autre valeur non vide du tampon d'indicateur. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriétéPLOT_EMPTY_VALUE:

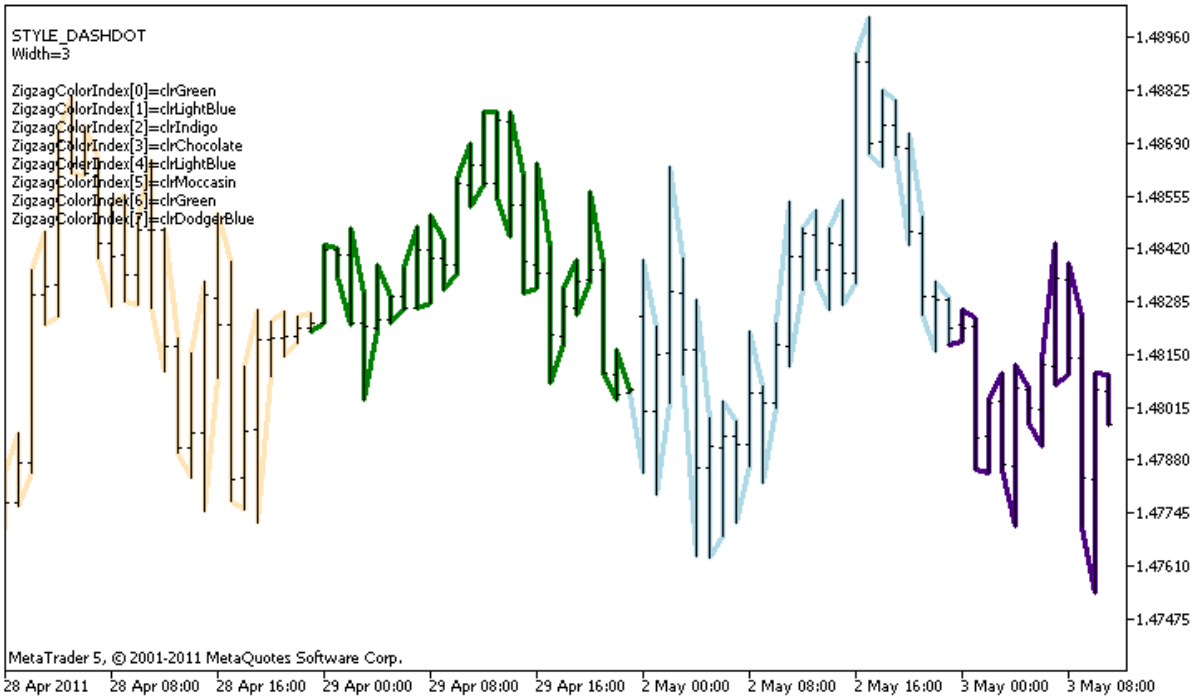
```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(l'index_de la construction_DRAW_COLOR_ZIGZAG,PLOT_EMPTY_VALUE,0)
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

Le nombre de tampons nécessaires pour la construction DRAW_COLOR_ZIGZAG — 3:

- deux tampons pour la conservation des valeurs des fins des sections du zigzag;
- un tampon pour stocker l'index de la couleur, par lequel la section est dessinée (il faut spécifier seulement pour les valeurs non vides).

L'exemple de l'indicateur dessinant la scie de long selon les prix High et Low. La couleur, l'épaisseur et le style de la ligne du zigzag se changent par hasard chaqueN ticks.



© 2000-2024, MetaQuotes Ltd.

Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_COLOR_ZIGZAG on spécifie 8 couleurs à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() la couleur est choisie par hasard parmi 14 couleurs stockées dans le tableau colors[].

Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_ZIGZAG.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_ZIGZAG"
#property description "Dessine la ligne brisée par les segments colorés, la couleur dé
#property description "La couleur, l'épaisseur et le style des segments se changent"
#property description " par hasard par chaque N ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plot Color_Zigzag
#property indicator_label1  "Color_Zigzag"
#property indicator_type1   DRAW_COLOR_ZIGZAG
//--- spécifions 8 couleurs pour colorer les sections (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- le paramètre input
input int      N=5;              // le nombre de ticks pour le changement
int            color_sections;
//--- les tampons des valeurs des fins des segments
double         Color_ZigzagBuffer1[];
double         Color_ZigzagBuffer2[];
//--- le tampon des index de la couleur des fins des segments
double         Color_ZigzagColors[];
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
```

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
//---- le nombre de couleurs pour colorier le zigzag
   color_sections=8;   //  regarder le commentaire pour la propriété #property indicat
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles les sections sont dessinées
      ChangeColors(colors,color_sections);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- la structure du temps sera nécessaire pour la réception du jour de la semaine de
   MqlDateTime dt;

//--- la position du commencement des calculs
   int start=0;
//---si l'indicateur a été calculé sur le tick précédent, nous commençons le calcul pa
   if(prev_calculated!=0) start=prev_calculated-1;
//--- la boucle des calculs
```

```
   for(int i=start;i<rates_total;i++)
     {
      //--- inscrirons le temps de l'ouverture de la barre dans la structure
      TimeToStruct(time[i],dt);

      //--- si le numéro de la barre est pair
      if(i%2==0)
        {
         //---  écrivons au 1-er tampon High, et au 2-ème - Low
         Color_ZigzagBuffer1[i]=high[i];
         Color_ZigzagBuffer2[i]=low[i];
         //--- la couleur du segment
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
      //--- le numéro de la barre est impair
      else
        {
         //--- remplissons la barre dans l'ordre inverse
         Color_ZigzagBuffer1[i]=low[i];
         Color_ZigzagBuffer2[i]=high[i];
         //--- la couleur du segment
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Change la couleur des segments du zigzag                         |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
```

```
      //--- inscrirons les couleurs
      comm=comm+StringFormat("ZigzagColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence des segments dans le zigzag                  |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés des Color_ZigZag
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   int size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_BARS

Le style DRAW_COLOR_BARS dessine les barres selon les valeurs de quatre tampons d'indicateur, qui contiennent les prix Open, High, Low et Close. Ce style est une version avancée du style DRAW_BARS et permet de spécifier la couleur de l'ensemble des couleurs d'avance prédéterminé pour chaque barre. Il est destiné à la création des indicateurs personnels en forme des barres, y compris dans une sous-fenêtre séparée du graphique et selon d'autres instruments financiers.

On peut spécifier la couleur des barres par les directives du compilateur ou dynamiquement à l'aide de la fonctionPlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de "vivifier" les indicateurs pour qu'ils changent l'aspect en fonction de la situation actuelle.

L'indicateur est dessiné seulement pour ces barres, pour lesquelles on a spécifié les valeurs non vides de **tous** les quatre tampons d'indicateur. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriété PLOT_EMPTY_VALUE:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(индекс_построения_DRAW_COLOR_BARS,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

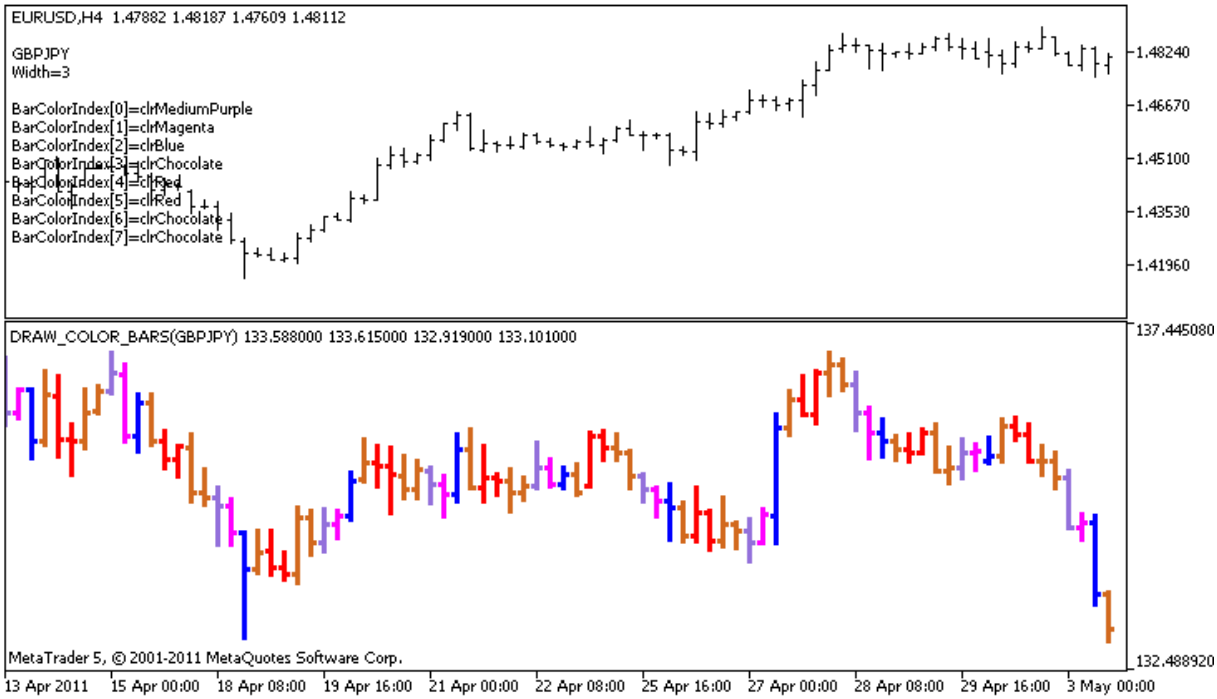
Le nombre de tampons nécessaires pour la construction DRAW_COLOR_BARS — 5:

- quatre tampons pour stocker les valeurs Open, High, Low et Close;
- un tampon pour stocker l'index de la couleur, par lequel la barre est dessinée (il faut spécifier seulement pour les barres dessinées).

Tous les tampons pour la construction doivent aller successivement un après l'autre en ordre donné: Open, High, Low, Close et le tampon de la couleur. Aucun des tampons de prix ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

L'exemple de l'indicateur dessinant les barres selon l'instrument financier indiqué dans la fenêtre séparée. La couleur des barres se change chaques **N** ticks par hasard. Le paramètre N est porté auxparamètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans les propriétés de l'indicateur).

© 2000-2024, MetaQuotes Ltd.



Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_COLOR_BARS on spécifie 8 couleurs à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() les couleurs sont choisies par hasard parmi 14 couleurs stockées dans le tableau colors[].

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_BARS.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_BARS"
#property description "Dessine dans la fenêtre séparée par les couleurs différentes le
#property description "La couleur et l'épaisseur et le symbole des barres se changent
#property description "par chaque N ticks"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorBars
#property indicator_label1  "ColorBars"
#property indicator_type1   DRAW_COLOR_BARS
//--- spécifions 8 couleurs pour colorer les barres (ils se trouvent dans un tableau s
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres input
```

```
input int      N=5;              // le nombre de ticks pour le changement de l'aspect
input int      bars=500;         // combien de barres montrer
input bool     messages=false;   // l'affichage des messages dans le journal "Experts"
//--- les tampons d'indicateur
double         ColorBarsBuffer1[];
double         ColorBarsBuffer2[];
double         ColorBarsBuffer3[];
double         ColorBarsBuffer4[];
double         ColorBarsColors[];
//--- le nom du symbole
string symbol;
int    bars_colors;
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorBarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorBarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorBarsBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorBarsColors,INDICATOR_COLOR_INDEX);
//---- le nombre de couleurs pour colorier les barres
   bars_colors=8;   //  à voir le commentaire pour la propriété #property indicator_co
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
```

```
//--- calculons les tics pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- choisirons un nouveau symbole de la fenêtre "Aperçu du marché"
      symbol=GetRandomSymbolName();
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles les barres sont dessinées
      ChangeColors(colors,bars_colors);
      int tries=0;
      //--- faisons 5 tentatives de remplir les tampons par les prix du symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
         //--- le compteur des appels de la fonction CopyFromSymbolToBuffers()
         tries++;
        }
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur par les prix                |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
  {
//--- copions les prix Open, High, Low et Close au tableau rates[]
   MqlRates rates[];
//--- le compteur des tentatives
   int attempts=0;
//--- combien on a été copié
   int copied=0;
//--- faisons 25 tentatives de recevoir la série temporelle selon le symbole nécessai
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si on n'a pas réussi à copier le nombre suffisant des barres
   if(copied!=bars)
     {
      //--- formons la chaîne du message
      string comm=StringFormat("On a réussi à recevoir seulement %d barres de %d barre
                               name,
                               copied,
```

```
                                bars
                                );
      //--- déduisons le message au commentaire sur une principale fenêtre du graphiqu
      Comment(comm);
      //--- déduisons les messages
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- établissons l'affichage du symbole
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS("+name+")");
     }
//--- initialisons les tampons par les valeurs vides
   ArrayInitialize(ColorBarsBuffer1,0.0);
   ArrayInitialize(ColorBarsBuffer2,0.0);
   ArrayInitialize(ColorBarsBuffer3,0.0);
   ArrayInitialize(ColorBarsBuffer4,0.0);

//--- copions les prix aux tampons
   for(int i=0;i<copied;i++)
     {
      //--- calculons l'index correspondant pour les tampons
      int buffer_index=total-copied+i;
      //--- inscrivons les prix aux tampons
      ColorBarsBuffer1[buffer_index]=rates[i].open;
      ColorBarsBuffer2[buffer_index]=rates[i].high;
      ColorBarsBuffer3[buffer_index]=rates[i].low;
      ColorBarsBuffer4[buffer_index]=rates[i].close;
      //---
      ColorBarsColors[buffer_index]=i%bar_colors;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Rend accidentellement le symbole du Market Watch                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- le nombre de symboles montrés dans la fenêtre "Aperçu du marché"
   int symbols=SymbolsTotal(true);
//--- la position du symbole dans la liste - le nombre accidentel de 0 jusqu'aux symbo
   int number=MathRand()%symbols;
//--- rendons le nom du symbole selon la position indiquée
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Change la couleur des segments du zigzag                         |
```

```
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("BarColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence des barres                                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés des barres
   string comm="";

//--- le bloc du changement de l'épaisseur de la barre
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- inscrirons le nom du symbole
   comm="\r\n"+symbol+comm;

//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_CANDLES

Le style DRAW_COLOR_CANDLES comme DRAW_CANDLES dessine les chandeliers japonais selon les valeurs de quatre tampons d'indicateur, qui contiennent les prix Open, High, Low et Close. Mais outre cela il permet de spécifier la couleur pour chaque bougie de l'ensemble donné. Pour cela on a ajouté le tampon de couleur spécial au style, qui garde les index des couleurs pour chaque barre. Il est destiné à la création des indicateurs personnels en forme des bougies, y compris dans une sous- fenêtre séparée du graphique et selon d'autres financiers financiers.

On peut spécifier le nombre de couleurs pour le coloriage des bougies par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de "vivifier" les indicateurs pour qu'ils changent l'aspect en fonction de la situation actuelle.

L'indicateur est dessiné seulement pour ces barres, pour lesquelles on a spécifié les valeurs non vides de quatre tampons d'indicateur pour garder les prix. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriétéPLOT_EMPTY_VALUE:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(l'index_de la construction_DRAW_COLOR_CANDLES,PLOT_EMPTY_VALUE,(
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

Le nombre de tampons nécessaires pour la construction DRAW_COLOR_CANDLES — 5:

- quatre tampons pour stocker les valeurs Open, High, Low et Close;
- le tampon pour stocker l'index de la couleur, par lequel la bougie est dessinée (il est logique de spécifier seulement les bougies dessinées).

Tous les tampons pour la construction doivent aller successivement un après l'autre en ordre donné: Open, High, Low, Close et le tampon de la couleur. Aucun des tampons de prix ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

L'exemple de l'indicateur dessinant les chandeliers japonaises selon l'instrument financier indiqué dans la fenêtre séparée. La couleur des chandeliers se change chaques**N** ticks par hasard. Le paramètre N est porté auxparamètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans les propriétés de l'indicateur).

© 2000-2024, MetaQuotes Ltd.

Prêtez attention que primordialement à la construction graphique **plot1**on spécifie la couleur à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() une nouvelle couleur est spécifiée par hasard de la liste d'avance préparée.

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_CANDLES.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_CANDLES."
#property description "Dessine dans la fenêtre séparée par les couleurs différentes le
#property description " "
#property description "La couleur et l'épaisseur et le symbole des bougies se changent
#property description "par hasard par chaque N ticks."

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
//--- spécifions 8 couleurs pour colorer les bougies (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```
//--- les paramètres input
input int      N=5;              // le nombre de ticks pour le changement de l'aspect
input int      bars=500;         // combien de bougies montrer
input bool     messages=false;   // l'affichage des messages dans le journal "Experts"
//--- les tampons d'indicateur
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- le nom du symbole
string symbol;
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- s'il y a trop peu de barres - terminons le travail avant terme
   if(bars<50)
     {
      Comment("Spécifiez un plus grand nombre de barres! Le travail de l'indicateur es
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorCandlesBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorCandlesBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorCandlesBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorCandlesBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---le nom du symbole, selon lequel les barres se dessinent
   symbol=_Symbol;
//--- établissons l'affichage du symbole
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
//---- le nombre de couleurs pour colorier les bougies
   candles_colors=8;     //  à voir le commentaire pour la propriété #property indicat
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- calculons les ticks pour le changement du style et de la couleur
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- choisirons un nouveau symbole de la fenêtre "Aperçu du marché"
      symbol=GetRandomSymbolName();
      //--- remplaçons l'aspect
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles les barres sont dessinées
      ChangeColors(colors,candles_colors);

      int tries=0;
      //--- faisons 5 tentatives de remplir les tampons plot1 par les prix du symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
            ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
            && tries<5)
        {
         //--- le compteur des appels de la fonction CopyFromSymbolToBuffers()
         tries++;
        }
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplit une bougie indiquée                                      |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
```

```
                              double &buff2[],
                              double &buff3[],
                              double &buff4[],
                              double &col_buffer[],
                              int    cndl_colors
                              )
  {
//--- copions les prix Open, High, Low et Close au tableau rates[]
   MqlRates rates[];
//--- le compteur des tentatives
   int attempts=0;
//--- combien on a été copié
   int copied=0;
//--- faisons 25 tentatives de recevoir la série temporelle selon le symbole nécessai
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si on n'a pas réussi à copier le nombre suffisant des barres
   if(copied!=bars)
     {
      //--- formons la chaîne du message
      string comm=StringFormat("On a réussi à recevoir seulement %d barres de %d barre
                               name,
                               copied,
                               bars
                               );
      //--- déduisons le message au commentaire sur une principale fenêtre du graphiqu
      Comment(comm);
      //--- déduisons les messages
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- établissons l'affichage du symbole
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
     }
//--- initialisons les tampons par les valeurs vides
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- copions les prix aux tampons sur chaque tick
   for(int i=0;i<copied;i++)
     {
```

```
      //--- calculons l'index correspondant pour les tampons
      int buffer_index=total-copied+i;
      //--- inscrivons les prix aux tampons
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
      //--- spécifions la couleur de la bougie
      int color_index=i%cndl_colors;
      col_buffer[buffer_index]=color_index;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Rend accidentellement le symbole du Market Watch                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- le nombre de symboles montrés dans la fenêtre "Aperçu du marché"
   int symbols=SymbolsTotal(true);
//--- la position du symbole dans la liste - le nombre accidentel de 0 jusqu'aux symbo
   int number=MathRand()%symbols;
//--- rendons le nom du symbole selon la position indiquée
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Change la couleur des segments des bougies                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("CandleColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
```

```
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence des bougies                                  |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés des bougies
   string comm="";
//--- inscrirons le nom du symbole
   comm="\r\n"+symbol+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

Le lien entre les propriétés de l'indicateur et les fonctions correspondantes

Chaque indicateur d'utilisateur possède la multitude de propriétés,dont la partie est obligatoire et se trouve toujours tout au début de la description. Ce sont les propriétés:

- l'indication de la fenêtre pour l'affichage de l'indicateur - indicator_separate_window ou indicator_chart_window;
- le nombre de tampons d'indicateur - indicator_buffers;
- le nombre de constructions graphiques, qui affiche l'indicateur - indicator_plots.

Mais il y a une autre partie des propriétés, que l'on peut spécifier dans les directives du préprocesseur, ainsi que dans les fonctions pour la création de l'indicateur d'utilisateur. Dans le tableau on énumère ces propriétés et les fonctions leur correspondant.

Les directives pour la propriété de la sous-fenêtre de l'indicateur	Fonctions du type IndicatorSet...()	Description de la propriété établie de sous- fenêtre
indicator_height	IndicatorSetInteger (INDICATOR_INDICATOR_HEIGHT, nHeight)	La valeur de hauteur fixe de sous- fenêtre
indicator_minimum	IndicatorSetDouble (INDICATOR_MINIMUM, dMaxValue)	La valeur minimale de l'axe vertical
indicator_maximum	IndicatorSetDouble (INDICATOR_MAXIMUM, dMinValue)	La valeur maximal de l'axe vertical
indicator_levelN	IndicatorSetDouble (INDICATOR_LEVELVALUE, N-1, nLevelValue)	La valeur de l'axe vertical pour le niveau le numéro N
il n'y a pas de directive du préprocesseur	IndicatorSetString (INDICATOR_LEVELTEXT, N-1, sLevelName)	Le nom du niveau affiché
indicator_levelcolor	IndicatorSetInteger (INDICATOR_LEVELCOLOR, N-1, nLevelColor)	La couleur pour l'affichage du niveau le numéro N
indicator_levelwidth	IndicatorSetInteger (INDICATOR_LEVELWIDTH, N-1, nLevelWidth)	L'épaisseur de la ligne pour l'affichage du niveau le numéro N
indicator_levelstyle	IndicatorSetInteger (INDICATOR_LEVELSTYLE, N-1, nLevelStyle)	Le style de la ligne pour l'affichage du niveau le numéro N
Directives pour la propriété des constructions graphiques	**Fonctions du type PlotIndexSet...()**	**Description de la propriété établie pour la construction graphique**

© 2000-2024, MetaQuotes Ltd.

Les directives pour la propriété de la sous-fenêtre de l'indicateur	Fonctions du type IndicatorSet...()	Description de la propriété établie de sous- fenêtre
indicator_labelN	PlotIndexSetString(N-1,PLOT_LABEL,sLabel)	Le nom bref pour la construction graphique le numéro N. Il est affiché dans la fenêtre DataWindow et dans l'aide émergeant à l'induction du curseur sur la construction
indicator_colorN	PlotIndexSetInteger(N-1, PLOT_LINE_COLOR, nColor)	La couleur de la ligne pour la construction graphique le numéro N
indicator_styleN	PlotIndexSetInteger(N-1, PLOT_LINE_STYLE, nType)	Le style de la ligne pour la construction graphique le numéro N
indicator_typeN	PlotIndexSetInteger(N-1, PLOT_DRAW_TYPE, nType)	Le type de la ligne pour la construction graphique le numéro N
indicator_widthN	PlotIndexSetInteger(N-1, PLOT_LINE_WIDTH, nWidth)	L'épaisseur de la ligne pour la construction graphique le numéro N
Propriétés communes de l'indicateur	**Fonctions du type IndicatorSet...()**	**Description**
il n'y a pas de directive du préprocesseur	IndicatorSetString (INDICATOR_SHORTNAME, sShortName)	Établit le nom confortable court de l'indicateur, qui sera affiché dans le répertoire des indicateurs (s'ouvre dans le terminal en appuyant sur **Ctrl+I**).
il n'y a pas de directive du préprocesseur	IndicatorSetInteger (INDICATOR_DIGITS, nDigits)	Établit l'exactitude demandée pour l'affichage des valeurs de l'indicateur - le nombre de signes après le point décimal
il n'y a pas de directive du préprocesseur	IndicatorSetInteger (INDICATOR_LEVELS, nLevels)	Spécifie le nombre de niveaux sur la fenêtre de l'indicateur
indicator_applied_price	Les fonctions sont absentes, la propriété s'établit seulement par la directive du préprocesseur.	Le type du prix par défaut, utilisé pour le calcul des valeurs de l'indicateur. Est indiqué en cas de nécessité seulement à l'utilisation de la fonction OnCalculate() du premier type. On peut aussi spécifier la valeur de la propriété du

© 2000-2024, MetaQuotes Ltd.

Les directives pour la propriété de la sous-fenêtre de l'indicateur	Fonctions du type IndicatorSet...()	Description de la propriété établie de sous- fenêtre
		dialogue des propriétés de l'indicateur sur l'onglet "Paramètres" - "Appliquer vers".

Il est nécessaire de noter, que le numérotage des niveaux et les constructions graphiques dans les termes du préprocesseur commence par l'unité, pendant que le numérotage des mêmes propriétés à l'utilisation des fonctions commence par le zéro, c'est-à-dire il faut indiquer la valeur, moins sur 1 que nous indiquerions avec l'utilisation #property.

Il y a quelques directives, pour lesquelles il n'y a pas fonctions correspondantes:

Directive	Description
indicator_chart_window	L'indicateur se montre dans une fenêtre principale
indicator_separate_window	L'indicateur se montre dans une sous-fenêtre séparée
indicator_buffers	Indique le nombre de tampons demandés d'indicateur
indicator_plots	Indique le nombre de constructions graphiques dans l'indicateur

© 2000-2024, MetaQuotes Ltd.

SetIndexBuffer

Attache le tampon d'indicateur indiquée avec le tableau unidimentionnel dynamique du type double.

```
bool  SetIndexBuffer(
   int                    index,          // index du tampon
   double                 buffer[],       // tableau
   ENUM_INDEXBUFFER_TYPE  data_type       // ce qui sera conservé
   );
```

Paramètres

index

[in] Le numéro du tampon d'indicateur. Le numérotage commence par 0. Le numéro doit être moins de la valeur déclarée à #property indicator_buffers.

buffer[]

[in] Le tableau déclaré dans le programme de l'indicateur d'utilisateur.

data_type

[in] Le type de données, qui se trouve dans le tableau d'indicateur. Par défaut INDICATOR_DATA (la valeur de l'indicateur calculé). Peut aussi accepter la valeur INDICATOR_COLOR_INDEX, alors ce tampon est destiné au stockage des index des couleurs pour le tampon précédent d'indicateur. On peut spécifier environ 64 couleurs dans la ligne #property indicator_colorN. La valeur INDICATOR_CALCULATIONS signifie que ce tampon participe aux calculs intermédiaires de l'indicateur et n'est pas destiné pour le dessin.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

Note

Après le binding le tableau dynamique *buffer[]* aura l'indexation comme dans les tableaux ordinaires, même si pour le tableau lié on établit préalablement l'indexation comme aux séries temporelles. S'il est nécessaire de changer l'ordre d'accès vers les éléments du tableau d'indicateur, il faut appliquer la fonction ArraySetAsSeries() après le binding du tableau par la fonction SetIndexBuffer(). Il faut prendre en attention qu'il est impossible de changer la grandeur pour les tableaux dynamiques, nommés comme les tampons d'indicateur par la fonction SetIndexBuffer(). Pour les tampons d'indicateur toutes les opérations du changement de la grandeur sont produites par le sous-système exécutant du terminal.

Exemple:

```
//+------------------------------------------------------------------+
//|                                              TestCopyBuffer1.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
input int                 shift=0;
//--- indicator buffers
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   if(AsSeries) ArraySetAsSeries(MABuffer,true);
   Print("Le tampon d'indicateur est la série temporelle = ",ArrayGetAsSeries(MABuffer
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Le tampon d'indicateur après SetIndexBuffer() est la série temporelle = ",
         ArrayGetAsSeries(MABuffer));

//--- changeons le numéro d'accès vers les éléments du tampon d'indicateur
   ArraySetAsSeries(MABuffer,AsSeries);

   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

```
                const long &volume[],
                const int &spread[])
  {
//--- copions les valeurs du glissant moyen au tampon MABuffer
   int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

   Print("MABuffer[0] = ",MABuffer[0]);// en fonction de la valeur AsSeries
                                       // recevrons la plus vieille valeur
                                       // ou sur la barre courante inachevée

//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Voir aussi

Les propriétés des indicateurs d'utilisateur, l'accès vers les séries temporelles et les indicateurs

© 2000-2024, MetaQuotes Ltd.

IndicatorSetDouble

Spécifie la valeur de la propriété correspondante de l'indicateur. La propriété de l'indicateur doit être du type double. Il y a 2 variantes de la fonction.

L'appel avec l'indication de l'identificateur de la propriété.

```
bool  IndicatorSetDouble(
   int     prop_id,           // identificateur
   double  prop_value         // valeur établie
   );
```

L'appel avec l'indication de l'identificateur et le modificateur de la propriété.

```
bool  IndicatorSetDouble(
   int     prop_id,           // identificateur
   int     prop_modifier,     // modificateur
   double  prop_value         // valeur établie
   )
```

Paramètres

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_CUSTOMIND_PROPERTY_DOUBLE.

prop_modifier

[in] Le modificateur de la propriété indiquée. Seulement les propriétés des niveaux demandent le modificateur. Le numérotage des niveaux commence par 0, c'est-à-dire pour spécifier la propriété du deuxième niveau il faut indiquer 1 (moins 1 que lors de l'utilisation de la directive du compilateur).

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

Note

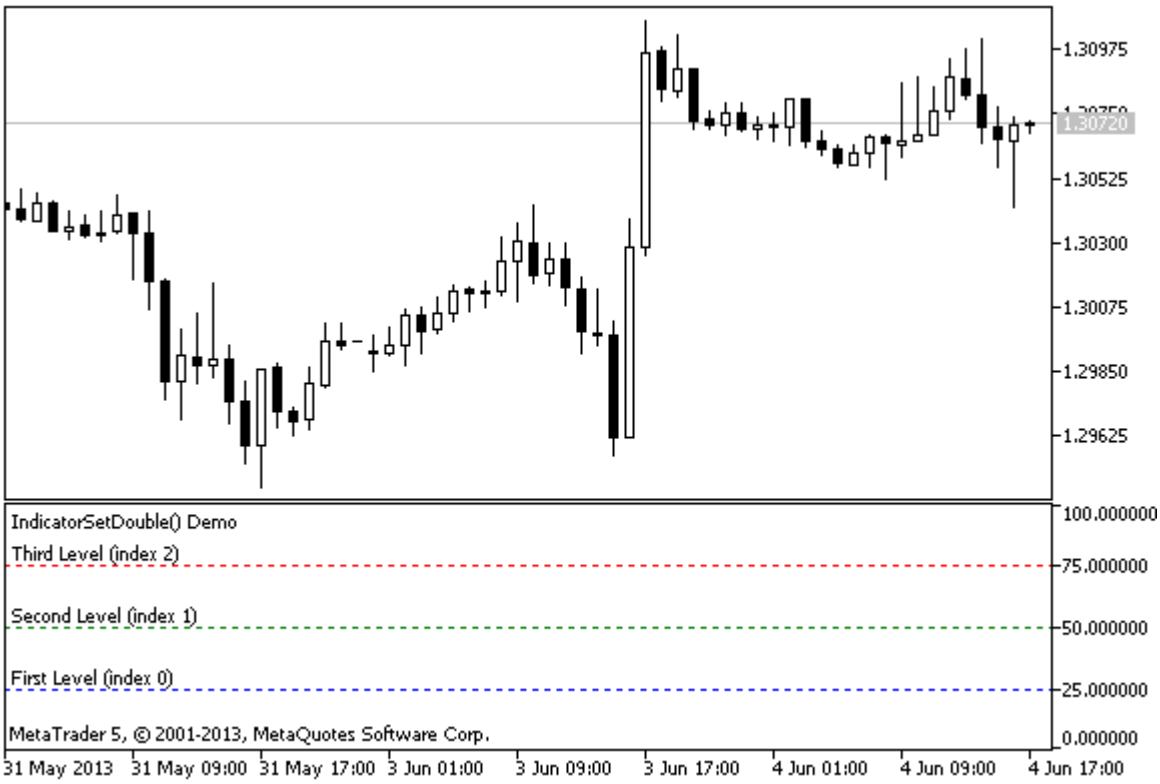
Le numérotage des propriétés (des modificateurs) à l'utilisation de la directive #property commence par 1 (un), pendant que la fonction utilise le numérotage par 0 (le zéro). Si le numéro du niveau était spécifié incorrectementl'affichage de l'indicateur peut se distinguer de celui qui est supposé.

Par exemple, on peut spécifier la valeur du premier niveau pour l'indicateur dans le sous- fenêtre séparé par deux moyens:

- property indicator_level**1** 50 - 1 est utilisé pour l'indication du numéro du niveau,
- IndicatorSetDouble(INDICATOR_LEVELVALUE, **0**, 50) - 0 est utilisé pour l'indication du premier niveau.

L'exemple: l'indicateur -"changeling", changeant les valeurs maximale et minimale de la fenêtre de l'indicateur, ainsi que les valeurs des niveaux, sur lesquels se trouvent les lignes horizontales.

© 2000-2024, MetaQuotes Ltd.




```
#property indicator_separate_window
//--- établirons les valeurs maximale et minimale pour la fenêtre de l'indicateur
#property indicator_minimum  0
#property indicator_maximum  100
//--- spécifions la démonstration de trois niveaux horizontaux dans la fenêtre séparée
#property indicator_level1 25
#property indicator_level2 50
#property indicator_level3 75
//--- établirons l'épaisseur des niveaux horizontaux
#property indicator_levelwidth 1
//--- établirons le style des niveaux horizontaux
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- spécifions les descriptions des niveaux horizontaux
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//---spécifions le nom court de l'indicateur
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetDouble() Demo");
//--- spécifions à chaque niveau son couleur
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,0,clrBlue);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,1,clrGreen);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,2,clrRed);
```

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
   static double level1=25,level2=50,level3=75;
   static double max=100,min=0, shift=100;
//--- comptons les ticks
   tick_counter++;
//--- faisons un coup sur chaque 10-ème tick
   if(tick_counter%10==0)
     {
      //--- retournerons le signe pour les valeurs du niveau
      level1=-level1;
      level2=-level2;
      level3=-level3;
      //--- retournerons le signe pour les valeurs maximales et minimales
      max-=shift;
      min-=shift;
      //--- retournerons la valeur du décalage
      shift=-shift;
      //--- spécifions les nouvelles valeurs des niveaux
      IndicatorSetDouble(INDICATOR_LEVELVALUE,0,level1);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,1,level2);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,2,level3);
      //--- spécifions les nouvelles valeurs maximales et minimales pour la fenêtre de
      Print("Set up max = ",max,",   min = ",min);
      IndicatorSetDouble(INDICATOR_MAXIMUM,max);
      IndicatorSetDouble(INDICATOR_MINIMUM,min);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Styles des indicateurs dans les exemples, Lien entre les propriétés de l'indicateur et les fonctions correspondantes, Styles du dessin

© 2000-2024, MetaQuotes Ltd.

IndicatorSetInteger

Spécifie la valeur de la propriété correspondante de l'indicateur. La propriété de l'indicateur doit être du type int ou color. Il y a 2 variantes de la fonction.

L'appel avec l'indication de l'identificateur de la propriété.

```
bool  IndicatorSetInteger(
   int  prop_id,           // identificateur
   int  prop_value         // valeur établie
   );
```

L'appel avec l'indication de l'identificateur et le modificateur de la propriété.

```
bool  IndicatorSetInteger(
   int  prop_id,           // identificateur
   int  prop_modifier,     // modificateur
   int  prop_value         // valeur établie
   )
```

Paramètres

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_CUSTOMIND_PROPERTY_INTEGER.

prop_modifier

[in] Le modificateur de la propriété indiquée. Seulement les propriétés des niveaux demandent le modificateur.

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

Note

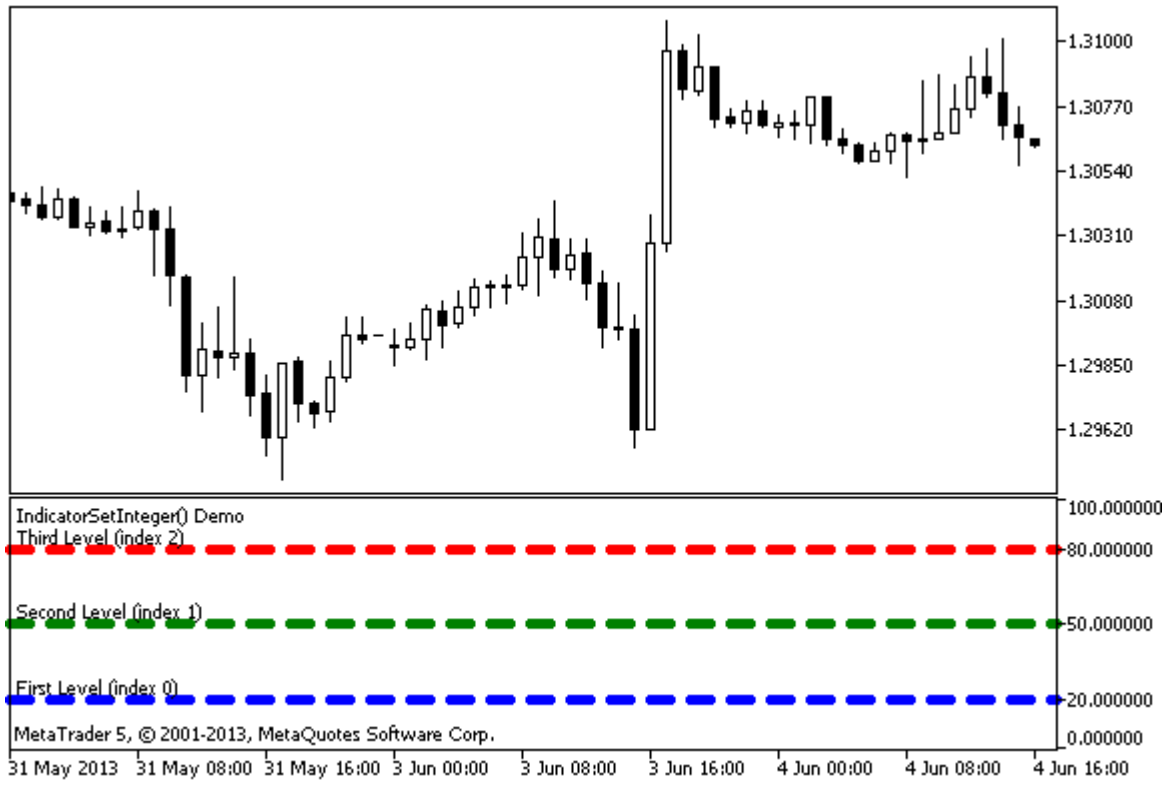
Le numérotage des propriétés (des modificateurs) à l'utilisation de la directive #property commence par 1 (une unité), pendant que la fonction utilise le numérotage par 0 (un zéro). Si le numéro du niveau était spécifié incorrectementl'affichage de l'indicateur peut se distinguer de celui qui est supposé.

Par exemple pour spécifier l'épaisseur de la ligne du premier niveau horizontal utilisez l'indice nul:

- IndicatorSetInteger(INDICATOR_LEVELWIDTH, **0**, 5) - on utilise l'indice 0 pour spécifier l'épaisseur de la ligne du premier niveau.

L'exemple: l'indicateur qui spécifie la couleur, le style et l'épaisseur des niveaux horizontaux de l'indicateur.

© 2000-2024, MetaQuotes Ltd.



© 2000-2024, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- spécifions la démonstration de trois niveaux horizontaux dans la fenêtre séparée
#property indicator_level1 20
#property indicator_level2 50
#property indicator_level3 80
//---établissons l'épaisseur des niveaux horizontaux
#property indicator_levelwidth 5
//--- établissons la couleur des niveaux horizontaux
#property indicator_levelcolor clrAliceBlue
//--- établissons le style des niveaux horizontaux
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- spécifions les descriptions des niveaux horizontaux
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- spécifions le nom court de l'indicateur
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetInteger() Demo");
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
//--- calculons les ticks
   tick_counter++;
//--- établissons les couleurs des niveaux horizontaux en fonction du compteur des tic
   ChangeLevelColor(0,tick_counter,3,6,10); // trois derniers paramètres basculent la
   ChangeLevelColor(1,tick_counter,3,6,8);
   ChangeLevelColor(2,tick_counter,4,7,9);
//--- changeons les styles des niveaux horizontaux
   ChangeLevelStyle(0,tick_counter);
   ChangeLevelStyle(1,tick_counter+5);
   ChangeLevelStyle(2,tick_counter+15);
//--- recevrons l'épaisseur de la ligne, comme le reste de la division entière du nomb
   int width=tick_counter%5;
//--- passerons selon tous les niveaux horizontaux et exposerons
   for(int l=0;l<3;l++)
      IndicatorSetInteger(INDICATOR_LEVELWIDTH,l,width+1);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| établissons la couleur de la ligne horizontale dans la fenêtre séparée de l'indica
```

```
//+------------------------------------------------------------------+
void ChangeLevelColor(int level,      // le numéro de la ligne horizontale
                      int tick_number,// le dividende, le nombre pour la réception du
                      int f_trigger,  // le premier diviseur du basculement de la coul
                      int s_trigger,  // le deuxième diviseur du basculement de la cou
                      int t_trigger)  // le troisième diviseur du basculement de la co
  {
   static color colors[3]={clrRed,clrBlue,clrGreen};
//--- l'indice de la couleur du tableau colors[]
   int index=-1;
//--- calculons le numéro de la couleur du tableau colors [] pour colorier la ligne ho
   if(tick_number%f_trigger==0)
      index=0;   // si le nombre tick_number se divise sans reste sur f_trigger
   if(tick_number%s_trigger==0)
      index=1;   // si le nombre tick_number se divise sans reste sur s_trigger
   if(tick_number%t_trigger==0)
      index=2;   // si le nombre tick_number se divise sans reste sur t_trigger
//--- si la couleur est définie, l'établissons
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELCOLOR,level,colors[index]);
//---
  }
//+------------------------------------------------------------------+
//| établissons le style de la ligne horizontale dans la fenêtre séparée de l'indicate
//+------------------------------------------------------------------+
void ChangeLevelStyle(int level,     // le numéro de la ligne horizontale
                      int tick_number// le nombre pour la réception du reste de la div
                      )
  {
//--- le tableau pour stocker les styles
   static ENUM_LINE_STYLE styles[5]=
     {STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//--- l'indice du style du tableau styles[]
   int index=-1;
//--- calculons le numéro du tableau styles[] pour établir le style de la ligne horizo
   if(tick_number%50==0)
      index=5;   // si tick_number se divise sans reste sur 50, le style STYLE_DASHDOT
   if(tick_number%40==0)
      index=4;   // ... le style STYLE_DASHDOT
   if(tick_number%30==0)
      index=3;   // ... STYLE_DOT
   if(tick_number%20==0)
      index=2;   // ... STYLE_DASH
   if(tick_number%10==0)
      index=1;   // ... STYLE_SOLID
//--- si le style est défini, l'établissons
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELSTYLE,level,styles[index]);
  }
```

Voir aussi

Propriétés des indicateurs d'utilisateurs, Propriétés des programmes (#property), Styles du dessin

© 2000-2024, MetaQuotes Ltd.

IndicatorSetString

Spécifie la valeur de la propriété correspondante de l'indicateur. La propriété de l'indicateur doit être du type string. Il y a 2 variantes de la fonction.

L'appel avec l'indication de l'identificateur de la propriété.

```
bool  IndicatorSetString(
   int     prop_id,           // identificateur
   string  prop_value         // valeur établie
   );
```

L'appel avec l'indication de l'identificateur et le modificateur de la propriété.

```
bool  IndicatorSetString(
   int     prop_id,           // identificateur
   int     prop_modifier,     // modificateur
   string  prop_value         // valeur établie
   )
```

Paramètres

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_CUSTOMIND_PROPERTY_STRING.

prop_modifier

[in] Le modificateur de la propriété indiquée. Seulement les propriétés des niveaux demandent le modificateur.

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

Note

Le numérotage des propriétés (des modificateurs) à l'utilisation de la directive #property commence par 1 (une unité), pendant que la fonction utilise le numérotage par 0 (un zéro). Si le numéro du niveau était spécifié incorrectementl'affichage de l'indicateur peut se distinguer de celui qui est supposé.

Par exemple pour spécifier la description du premier niveau horizontal utilisez l'indice nul:

- IndicatorSetString(INDICATOR_LEVELTEXT, **0**, "First Level") - on utilise l'indice 0 pour spécifier la tâche de la description de texte du premier niveau.

L'exemple: l'indicateur qui établie les signatures vers les niveaux horizontaux de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- spécifions la démonstration de trois niveaux horizontaux dans la fenêtre séparée
#property indicator_level1 30
#property indicator_level2 50
#property indicator_level3 70
//---établissons la couleur des niveaux horizontaux
#property indicator_levelcolor clrRed
//--- établissons le style des niveaux horizontaux
#property indicator_levelstyle STYLE_SOLID
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- spécifions les descriptions des niveaux horizontaux
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//---spécifions le nom court de l'indicateur
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetString() Demo");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
```

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---

//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Propriétés des indicateurs d'utilisateurs, Propriétés des programmes (#property)

© 2000-2024, MetaQuotes Ltd.

PlotIndexSetDouble

Spécifie la valeur de la propriété correspondante de la ligne correspondante de l'indicateur. La propriété de l'indicateur doit être du type double.

```
bool  PlotIndexSetDouble(
   int     plot_index,     // index du style graphique
   int     prop_id,        // identificateur de la propriété
   double  prop_value      // valeur établie
   );
```

Paramètres

plot_index

[in] L'index de la construction graphique

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_PLOT_PROPERTY_DOUBLE.

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

© 2000-2024, MetaQuotes Ltd.

PlotIndexSetInteger

Spécifie la valeur de la propriété correspondante de la ligne correspondante de l'indicateur. La propriété de l'indicateur doit être du typeint, char, bool ou color. Il y a 2 variantes de la fonction.

L'appel avec l'indication de l'identificateur de la propriété.

```
bool  PlotIndexSetInteger(
   int  plot_index,        // index du style graphique
   int  prop_id,           // identificateur de la propriété
   int  prop_value         // valeur établie
   );
```

L'appel avec l'indication de l'identificateur et le modificateur de la propriété.

```
bool  PlotIndexSetInteger(
   int  plot_index,        // index du style graphique
   int  prop_id,           // identificateur de la propriété
   int  prop_modifier,     // modificateur de la propriété
   int  prop_value         // valeur établie
   )
```

Paramètres

plot_index

[in] L'index de la construction graphique

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Le modificateur de la propriété indiquée. Seulement les propriétés des index des couleurs demandent le modificateur.

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

Exemple: l'indicateur qui dessine la ligne tricolore. Le schéma coloré change dans chaque 5 ticks.

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plot ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrGreen,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- indicator buffers
double         ColorLineBuffer[];
double         ColorBuffer[];
int            MA_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- get MA handle
   MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| get color index                                                  |
//+------------------------------------------------------------------+
```

```
int getIndexOfColor(int i)
  {
   int j=i%300;
   if(j<100) return(0);// first index
   if(j<200) return(1);// second index
   return(2); // third index
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   static int ticks=0,modified=0;
   int limit;
//--- first calculation or number of bars was changed
   if(prev_calculated==0)
     {
      //--- copy values of MA into indicator buffer ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);// copying failed - throw away
      //--- now set line color for every bar
      for(int i=0;i<rates_total;i++)
         ColorBuffer[i]=getIndexOfColor(i);
     }
   else
     {
      //--- copy values of MA into indicator buffer ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);

      ticks++;// ticks counting
      if(ticks>=5)//it's time to change color scheme
        {
         ticks=0; // reset counter
         modified++; // counter of color changes
         if(modified>=3)modified=0;// reset counter
         ResetLastError();
         switch(modified)
           {
```

```
            case 0:// first color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
               Print("Color scheme "+modified);
               break;
            case 1:// second color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
               Print("Color scheme "+modified);
               break;
            default:// third color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
               Print("Color scheme "+modified);
            }
         }
      else
         {
          //--- set start position
          limit=prev_calculated-1;
          //--- now we set line color for every bar
          for(int i=limit;i<rates_total;i++)
             ColorBuffer[i]=getIndexOfColor(i);
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

PlotIndexSetString

Spécifie la valeur de la propriété correspondante de la ligne correspondante de l'indicateur. La propriété de l'indicateur doit être du type string.

```
bool  PlotIndexSetString(
   int     plot_index,     // index du style graphique
   int     prop_id,        // identificateur de la propriété
   string  prop_value      // valeur établie
   );
```

Paramètres

plot_index

[in] L'index de la construction graphique

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_PLOT_PROPERTY_STRING.

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

© 2000-2024, MetaQuotes Ltd.

PlotIndexGetInteger

Spécifie la valeur de la propriété correspondante de la ligne correspondante de l'indicateur. La propriété de l'indicateur doit être du type int, color, bool ou char. Il y a 2 variantes de la fonction.

L'appel avec l'indication de l'identificateur de la propriété.

```
int  PlotIndexGetInteger(
   int  plot_index,        // index du style graphique
   int  prop_id,           // identificateur de la propriété
   );
```

L'appel avec l'indication de l'identificateur et le modificateur de la propriété.

```
int  PlotIndexGetInteger(
   int  plot_index,        // index du style graphique
   int  prop_id,           // identificateur de la propriété
   int  prop_modifier      // identificateur de la propriété
   )
```

Paramètres

plot_index

[in] L'index de la construction graphique

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Le modificateur de la propriété indiquée. Seulement les propriétés des index des couleurs demandent le modificateur.

Note

La fonction est destinée à l'extrait des ajustements du dessin de la ligne correspondante de l'indicateur. La fonction travaille avec la fonction PlotIndexSetInteger pour le copiage des propriétés du dessin d'une ligne à l'autre.

Exemple: l'indicateur qui colorie des bougies selon le jour de la semaine. Les couleurs pour chaque jour sont spécifiés d'une manière de programme.

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         OpenBuffer[];
double         HighBuffer[];
double         LowBuffer[];
double         CloseBuffer[];
double         ColorCandlesColors[];
color          ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                              clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,OpenBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,HighBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LowBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,CloseBuffer,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- set number of colors in color buffer
```

© 2000-2024, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- set colors for color buffer
   for(int i=1;i<6;i++)
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- set accuracy
   IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
   printf("We have %u colors of days",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   int i;
   MqlDateTime t;
//----
   if(prev_calculated==0) i=0;
   else i=prev_calculated-1;
//----
   while(i<rates_total)
     {
      OpenBuffer[i]=open[i];
      HighBuffer[i]=high[i];
      LowBuffer[i]=low[i];
      CloseBuffer[i]=close[i];
      //--- set color for every candle
      TimeToStruct(time[i],t);
      ColorCandlesColors[i]=t.day_of_week;
      //---
      i++;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

Les objets graphiques

Le groupe des fonctions destinées au travail avec les objets graphiques, se rapportant à n'importe quel graphique indiqué. On ne peut pas utiliser ces fonctions dans les indicateurs.

Les fonctions qui définissent les propriétés des objets graphiques, ainsi que les opérations de la création de ObjectCreate() et de déplacement ObjectMove() des objets sur le graphique, servent en réalité à envoyer des commandes au graphique. À l'exécution réussie de ces fonctions la commande se trouve dans une file d'attente des événements du graphique. Le changement visuel des propriétés des objets graphiques s'est produit en train du traitement de la file d'attente des événements du graphique donné.

Pour cette raison il ne faut pas attendre la mise à jour visuelle immédiate des objets graphiques après l'appel des fonctions données. En général, la mise à jour des objets graphiques s'est produite par le terminal automatiquement selon les événements du changement - l'entrée de la nouvelle cotation, le changement de la taille de la fenêtre du graphique etc.

Pour la mise à jour forcée des objets graphiques utilisez la commande pour redessiner le graphique ChartRedraw().

Fonction	Action
ObjectCreate	Crée l'objet du type spécifié sur le graphique indiqué
ObjectName	Rend le nom de l'objet du type correspondant dans le graphique indiqué (indiqué dans la sous- fenêtre du graphique)
ObjectDelete	Supprime l'objet avec le nom indiqué du graphique indiqué (de la sous-fenêtre de graphique indiquée)
ObjectsDeleteAll	Supprime tous les objets du type indiqué du graphique indiqué (de la sous-fenêtre de graphique indiquée)
ObjectFind	Cherche l'objet selon le nom avec l'identificateur indiqué
ObjectGetTimeByValue	Rend la valeur du temps pour la valeur indiquée du prix de l'objet
ObjectGetValueByTime	Rend la valeur de prix de l'objet pour le temps indiqué
ObjectMove	Change les coordonnées du point indiqué du rattachement de l'objet
ObjectsTotal	Rend le nombre d'objets du type indiqué dans le graphique indiqué (la sous-fenêtre de graphique indiquée)
ObjectGetDouble	Rend la valeur du type double de la propriété correspondante de l'objet
ObjectGetInteger	Rend la valeur entière de la propriété correspondante de l'objet
ObjectGetString	Rend la valeur du type string de la propriété correspondante de l'objet
ObjectSetDouble	Met la valeur de la propriété correspondante de l'objet
ObjectSetInteger	Met la valeur de la propriété correspondante de l'objet
ObjectSetString	Met la valeur de la propriété correspondante de l'objet

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
TextSetFont	Établit la fonte pour la sortie du texte par les méthodes du dessin (on utilise par défaut la fonte Arial 20)
TextOut	Déduit le texte au tableau d'utilisateur (le tampon) destiné à la création de la ressourcegraphique
TextGetSize	Rend la largeur et la hauteur de la ligne aux réglages courants de la fonte

Chaque objet graphique doit avoir le nom, unique dans la limite d'un graphique, y compris ses sous-fenêtres. Le changement du nom de l'objet graphique forme deux événements: premier est un événement de l'éloignement de l'objet avec un vieux nom, et deuxième – l'événement de la création de l'objet graphique avec un nouveau nom.

Après la création de l'objet ou la modification des propriétés de l'objet il est recommandé d'appeler la fonction ChartRedraw(), qui rend au terminal l'ordre pour le dessin forcé du graphique (et de tous les objets visibles sur lui).

© 2000-2024, MetaQuotes Ltd.

ObjectCreate

Crée l'objet avec le nom indiqué, le type et les coordonnées initiales de la sous-fenêtre indiquée du graphique. On peut indiquer environ 30 coordonnées pendant la création.

```
bool  ObjectCreate(
   long         chart_id,       // identificateur du graphique
   string       name,           // nom de l'objet
   ENUM_OBJECT  type,           // type de l'objet
   int          sub_window,     // index de la fenêtre
   datetime     time1,          // heure du premier point d'ancrage
   double       price1,         // prix du premier point d'ancrage
   ...
   datetime     timeN=0,        // heure du N-ème point d'ancrage
   double       priceN=0,       // prix du N-ème point d'ancrage
   ...
   datetime     time30=0,       // heure du 30-ème point d'ancrage
   double       price30=0       // prix du 30-ème point d'ancrage
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet. Le nom doit être unique dans la limite d'un graphique, y compris ses sous-fenêtres.

type

[in] Le type de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique. La sous-fenêtre indiquée doit exister, dans le cas contraire la fonction retourne false.

time1

[in] La coordonnée temporelle du premier rattachement.

price1

[in] La coordonnée de prix du premier point d'ancrage.

timeN=0

[in] La coordonnée temporelle du N-ème point d'ancrage.

priceN=0

[in] La coordonnée de prix du N-ème point d'ancrage.

time30=0

[in] La coordonnée temporelle du trentième point d'ancrage.

price30=0

© 2000-2024, MetaQuotes Ltd.

[in] La coordonnée de prix du trentième point d'ancrage.

Valeur de Retour

La fonction retourne true si la commande a été ajoutée avec succès dans la queue du graphique spécifié, ou sinon false. Si un objet a déjà été créé, une tentative est faite pour changer ses coordonnées.

Note

Un appel asynchrone est toujours effectué pour ObjectCreate(), c'est pourquoi la fonction ne retourne que le résultat de l'ajout de la commande dans la queue du graphique. Dans ce cas, true signifie que la commande a été placée dans queue avec succès, mais le rsultat de son exécution est inconnu.

Pour vérifier le résultat de l'exécution de la commande, vous pouvez utiliser la fonction ObjectFind() ou toute autre fonction qui accède aux propriétés de l'objet, telle que ObjectGetXXX. Vous devriez cependant conserver à l'esprit que ce genre de fonctions sont ajoutées à la fin de la queue de ce graphique, et qu'elles attendent le résultat de l'exécution (en raison de l'appel synchrone), et peuvent donc être consommatrices en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

Le nom d'un objet ne doit pas excéder 63 caractères.

La numérotation des sous-fenêtres du graphique (si sur le graphique il y a des sous-fenêtres avec des indicateurs) commence par 1. La fenêtre principale du graphique existe toujours et elle a l'index 0.

Une grande quantité de points d'ancrage (jusqu'à 30) est prévue pour une utilisation future. La limite de 30 points d'ancrage possibles pour les objets graphiques est dûe au fait que le nombre de paramètres lors de l'appel de la fonction ne doit pas excéder 64.

Lors du changement de nom d'un objet graphique, deux événements sont générés, qui peuvent être traité dans les experts ou dans les indicateurs avec la fonction OnChartEvent() :

- l'événement de la suppression de l'objet avec le nom précédent ;
- l'événement de la création de l'objet graphique avec un nouveau nom.

Pour créer chacun des types des objets, il faut spécifier la quantité de points d'ancrage :

Identificateur	Description	Points d'ancrage
OBJ_VLINE	Ligne verticale	Un point d'ancrage. Seule la coordonnée du l'axe du temps est utilisée.
OBJ_HLINE	Ligne horizontale	Un point d'ancrage. Seule la coordonnée du l'axe des prix est utilisée.
OBJ_TREND	Ligne de tendance	Deux points d'ancrage
OBJ_TRENDBYANGLE	Ligne de tendance selon un angle	Deux points d'ancrage
OBJ_CYCLES	Lignes de cycles	Deux points d'ancrage

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Points d'ancrage
OBJ_ARROWED_LINE	Ligne fléchée	Deux points d'ancrage
OBJ_CHANNEL	Canal équidistant	Trois points d'ancrage
OBJ_STDDEVCHANNEL	Canal de divergence standard	Deux points d'ancrage
OBJ_REGRESSION	Canal de régression linéaire	Deux points d'ancrage
OBJ_PITCHFORK	Fourchette d'Andrews	Trois points d'ancrage
OBJ_GANNLINE	Ligne de Gann	Deux points d'ancrage
OBJ_GANNFAN	Evantail de Gann	Deux points d'ancrage
OBJ_GANNGRID	Grille de Gann	Deux points d'ancrage
OBJ_FIBO	Niveaux de Fibonacci	Deux points d'ancrage
OBJ_FIBOTIMES	Périodes de Fibonacci	Deux points d'ancrage
OBJ_FIBOFAN	Eventail de Fibonacci	Deux points d'ancrage
OBJ_FIBOARC	Arcs de Fibonacci	Deux points d'ancrage
OBJ_FIBOCHANNEL	Canal de Fibonacci	Trois points d'ancrage
OBJ_EXPANSION	Extension de Fibonacci	Trois points d'ancrage
OBJ_ELLIOTWAVE5	Vague d'impulsion d'Elliot	Cinq points d'ancrage.
OBJ_ELLIOTWAVE3	Vague de correction d'Elliot	Trois points d'ancrage
OBJ_RECTANGLE	Rectangle	Deux points d'ancrage
OBJ_TRIANGLE	Triangle	Trois points d'ancrage
OBJ_ELLIPSE	Ellipse	Trois points d'ancrage
OBJ_ARROW_THUMB_UP	Signe "pouce en haut"	Un point d'ancrage
OBJ_ARROW_THUMB_DOWN	Signe "pouce en bas"	Un point d'ancrage
OBJ_ARROW_UP	Flèche vers le haut	Un point d'ancrage
OBJ_ARROW_DOWN	Flèche vers le bas	Un point d'ancrage
OBJ_ARROW_STOP	Signe "Stop"	Un point d'ancrage
OBJ_ARROW_CHECK	Signe "Coche"	Un point d'ancrage
OBJ_ARROW_LEFT_PRICE	Etiquette de prix à gauche	Un point d'ancrage
OBJ_ARROW_RIGHT_PRICE	Etiquette de prix à droite	Un point d'ancrage
OBJ_ARROW_BUY	Signe "Achat"	Un point d'ancrage
OBJ_ARROW_SELL	Signe "Vente"	Un point d'ancrage
OBJ_ARROW	Flèche	Un point d'ancrage

© 2000-2024, MetaQuotes Ltd.

Identificateur	Description	Points d'ancrage
OBJ_TEXT	Objet "Texte"	Un point d'ancrage
OBJ_LABEL	Etiquette	La position est définie par ses propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.
OBJ_BUTTON	Bouton	La position est définie par ses propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.
OBJ_CHART	Graphique	La position est définie par ses propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.
OBJ_BITMAP	Dessin	Un point d'ancrage
OBJ_BITMAP_LABELl	Etiquette bitmap	La position est définie par ses propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.
OBJ_EDIT	Zone d'édition	La position est définie par ses propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.
OBJ_EVENT	Objet "Événement", correspondant à un événement dans le calendrier économique	Un point d'ancrage. Seule la coordonnée du l'axe du temps est utilisée.
OBJ_RECTANGLE_LABEL	Objet "Etiquette rectangle" pour la création et la présentation de l'interface utilisateur graphique.	La position est définie par ses propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.

© 2000-2024, MetaQuotes Ltd.

ObjectName

Rend le nom de l'objet correspondant au chart indiqué, dans la sous-fenêtre indiquée de chart indiqué, du type indiqué.

```
string  ObjectName(
   long  chart_id,            // identificateur du graphique
   int   pos,                 // numéro dans la liste des objets
   int   sub_window=-1,       // numéro de la fenêtre
   int   type=-1              // type de l'objet
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

pos

[in] Le numéro consécutif de l'objet selon le filtre indiqué selon le numéro de la sous-fenêtre et du type.

sub_window=-1

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie une fenêtre principale du graphique, -1 signifie tous les sous-fenêtres du graphique, y compris une fenêtre principale.

type=-1

[in] Le type de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT. -1 signifie tous les types.

Valeur de Retour

Le nom de l'objet en cas de succès.

Note

La fonction utilise un appel synchrone, ce qui signifie que la fonction attend la fin de l'exécution de toutes les commandes présentes dans la queue de ce graphique avant cet appel. C'est pourquoi cette fonction peut être consommatrice en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

Lors du changement de nom d'un objet graphique, deux événements sont générés, qui peuvent être traité dans les experts ou dans les indicateurs avec la fonction OnChartEvent() :

- l'événement de la suppression de l'objet avec le nom précédent ;
- l'événement de la création de l'objet graphique avec un nouveau nom.

© 2000-2024, MetaQuotes Ltd.

ObjectDelete

Supprime l'objet avec le nom indiqué du graphique indiqué.

```
bool  ObjectDelete(
   long    chart_id,      // identificateur du graphique
   string  name           // nom de l'objet
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet supprimé.

Valeur de Retour

La fonction retourne true si la commande a été ajoutée avec succès dans la queue du graphique spécifié, ou sinon false.

Note

Un appel asynchrone est toujours effectué pour ObjectDelete(), c'est pourquoi la fonction ne retourne que le résultat de l'ajout de la commande dans la queue du graphique. Dans ce cas, true signifie que la commande a été placée dans queue avec succès, mais le rsultat de son exécution est inconnu.

Pour vérifier le résultat de l'exécution de la commande, vous pouvez utiliser la fonction ObjectFind() ou toute autre fonction qui accède aux propriétés de l'objet, telle que ObjectGetXXX. Vous devriez cependant conserver à l'esprit que ce genre de fonctions sont ajoutées à la fin de la queue de ce graphique, et qu'elles attendent le résultat de l'exécution (en raison de l'appel synchrone), et peuvent donc être consommatrices en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

Lors du changement de nom d'un objet graphique, deux événements sont générés, qui peuvent être traité dans les experts ou dans les indicateurs avec la fonction OnChartEvent() :

- l'événement de la suppression de l'objet avec le nom précédent ;
- l'événement de la création de l'objet graphique avec un nouveau nom.

© 2000-2024, MetaQuotes Ltd.

ObjectsDeleteAll

Supprime tous les objets du graphique indiqué, dans la sous-fenêtre indiquée du graphique indiqué, du type indiqué.

```
int  ObjectsDeleteAll(
   long  chart_id,           // identificateur du graphique
   int   sub_window=-1,      // index de la fenêtre
   int   type=-1             // type de l'objet à effacer
   );
```

Supprime tous les objets du type spécifié et utilisant le préfixe indiqué dans leur nom.

```
int  ObjectsDeleteAll(
   long           chart_id,  // identifiant du graphique
   const string     prefix,  // préfixe dans le nom de l'objet
   int        sub_window=-1, // indice de la fenêtre
   int       object_type=-1  // type des objets
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

prefix

[in] Préfixe dans les noms des objets. Tous les objets dont le nom commence avec cet ensemble de caractères seront supprimés du graphique. Vous pouvez spécifier le préfixe sous la forme 'nom' ou 'nom*' - les 2 variantes fonctionneront de la même façon. Si une chaîne vide est spécifiée comme préfixe, tous les objets avec tous les noms possibles seront supprimés.

sub_window=-1

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie une fenêtre principale du graphique, -1 signifie tous les sous-fenêtres du graphique, y compris une fenêtre principale.

type=-1

[in] Le type de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT. -1 signifie tous les types.

Valeur de Retour

Retourne le nombre d'objets supprimés. Pour obtenir des informations supplémentaires sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

La fonction utilise un appel synchrone, ce qui signifie que la fonction attend la fin de l'exécution de toutes les commandes présentes dans la queue de ce graphique avant cet appel. C'est pourquoi cette fonction peut être consommatrice en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

© 2000-2024, MetaQuotes Ltd.

ObjectFind

Recherche l'objet ayant le nom indiqué dans le graphique indiqué.

```
int  ObjectFind(
   long    chart_id,      // identificateur du graphique
   string  name           // nom de l'objet
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet cherché.

Valeur de Retour

En cas du succès la fonction rend le numéro de la sous-fenêtre (0 signifie une principale fenêtre du graphique), dans laquelle il y a un objet trouvé. Si l'objet n'est pas trouvé, la fonction retourne un nombre négatif. Pour obtenir des informations supplémentaires sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

La fonction utilise un appel synchrone, ce qui signifie que la fonction attend la fin de l'exécution de toutes les commandes présentes dans la queue de ce graphique avant cet appel. C'est pourquoi cette fonction peut être consommatrice en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

Lors du changement de nom d'un objet graphique, deux événements sont générés, qui peuvent être traité dans les experts ou dans les indicateurs avec la fonction OnChartEvent() :

- l'événement de la suppression de l'objet avec le nom précédent ;
- l'événement de la création de l'objet graphique avec un nouveau nom.

© 2000-2024, MetaQuotes Ltd.

ObjectGetTimeByValue

Retourne la valeur du temps pour l'objet indiqué avec son prix.

```
datetime  ObjectGetTimeByValue(
   long    chart_id,      // identificateur du graphique
   string  name,          // nom de l'objet
   double  value,         // prix
   int     line_id        // ligne
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

value

[in] La valeur du prix.

line_id

[in] L'identificateur de la ligne.

Valeur de Retour

La valeur du temps pour la valeur indiquée du prix de l'objet indiqué.

Note

La fonction utilise un appel synchrone, ce qui signifie que la fonction attend la fin de l'exécution de toutes les commandes présentes dans la queue de ce graphique avant cet appel. C'est pourquoi cette fonction peut être consommatrice en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

Puisqu'un objet à une coordonnée de prix peut avoir plusieurs valeurs, il est nécessaire d'indiquer le numéro de la ligne. Cette fonction est employée seulement pour les objets suivants :

- La ligne de tendance (OBJ_TREND)
- La ligne de tendance selon un angle (OBJ_TRENDBYANGLE)
- Le ligne de Gann (OBJ_GANNLINE)
- Le canal équidistant (OBJ_CHANNEL) - 2 lignes
- Le canal de régression linéaire (OBJ_REGRESSION) - 3 lignes
- Le canal d'écart type (OBJ_STDDEVCHANNEL) - 3 lignes
- La ligne flêchée (OBJ_ARROWED_LINE)

Voir aussi

Les types des objets

© 2000-2024, MetaQuotes Ltd.

ObjectGetValueByTime

Retourne la valeur du prix pour le temps indiqué de l'objet indiqué.

```
double  ObjectGetValueByTime(
   long      chart_id,     // identificateur du graphique
   string    name,         // nom de l'objet
   datetime  time,         // temps
   int       line_id       // ligne
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

time

[in] La valeur du temps.

line_id

[in] L'identificateur de la ligne.

Valeur de Retour

La valeur du prix pour le temps indiqué de l'objet indiqué.

Note

La fonction utilise un appel synchrone, ce qui signifie que la fonction attend la fin de l'exécution de toutes les commandes présentes dans la queue de ce graphique avant cet appel. C'est pourquoi cette fonction peut être consommatrice en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

Puisqu'un objet à une coordonnée de prix peut avoir plusieurs valeurs, il est nécessaire d'indiquer le numéro de la ligne. Cette fonction est employée seulement pour les objets suivants :

- La ligne de tendance (OBJ_TREND)
- La ligne de tendance selon un angle (OBJ_TRENDBYANGLE)
- Le ligne de Gann (OBJ_GANNLINE)
- Le canal équidistant (OBJ_CHANNEL) - 2 lignes
- Le canal de régression linéaire (OBJ_REGRESSION) - 3 lignes
- Le canal d'écart type (OBJ_STDDEVCHANNEL) - 3 lignes
- La ligne flêchée (OBJ_ARROWED_LINE)

Voir aussi

Les types des objets

© 2000-2024, MetaQuotes Ltd.

ObjectMove

Change les coordonnées du point d'ancrage indiqué de l'objet.

```
bool  ObjectMove(
   long      chart_id,        // identificateur du graphique
   string    name,            // nom de l'objet
   int       point_index,     // numéro du point d'ancrage
   datetime  time,            // temps
   double    price            // prix
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

point_index

[in] Le numéro du point d'ancrage. Le nombre de points d'ancrage dépend du type de l'objet.

time

[in] La coordonnée temporaire du point d'ancrage indiqué.

price

[in] La coordonnée de prix du point d'ancrage indiqué.

Valeur de Retour

La fonction retourne true si la commande a été ajoutée avec succès dans la queue du graphique spécifié, ou sinon false.

Note

Un appel asynchrone est toujours effectué pour ObjectMove(), c'est pourquoi la fonction ne retourne que le résultat de l'ajout de la commande dans la queue du graphique. Dans ce cas, true signifie que la commande a été placée dans queue avec succès, mais le rsultat de son exécution est inconnu.

Pour vérifier le résultat de l'exécution de la commande, vous pouvez utiliser une fonction demandant les propriétés de l'objet, telle que ObjectGetXXX. Vous devriez cependant conserver à l'esprit que ce genre de fonctions sont ajoutées à la fin de la queue de ce graphique, et qu'elles attendent le résultat de l'exécution (en raison de l'appel synchrone), et peuvent donc être consommatrices en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

© 2000-2024, MetaQuotes Ltd.

ObjectsTotal

Rend le nom de l'objet correspondant au chart indiqué, dans la sous-fenêtre indiquée de chart indiqué, du type indiqué.

```
int  ObjectsTotal(
   long  chart_id,            // identificateur du graphique
   int   sub_window=-1,       // index de la fenêtre
   int   type=-1              // type de l'objet
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

sub_window=-1

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique, -1 signifie toutes les sous-fenêtres du graphique, y compris la fenêtre principale.

type=-1

[in] Le type de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT. -1 signifie tous les types.

Valeur de Retour

Le nombre d'objets.

Note

La fonction utilise un appel synchrone, ce qui signifie que la fonction attend la fin de l'exécution de toutes les commandes présentes dans la queue de ce graphique avant cet appel. C'est pourquoi cette fonction peut être consommatrice en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

© 2000-2024, MetaQuotes Ltd.

ObjectSetDouble

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type double. Il y a 2 variantes de la fonction.

Version sans modificateur

```
bool  ObjectSetDouble(
   long                          chart_id,          // identificateur du graphique
   string                        name,              // nom
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // propriété
   double                        prop_value         // valeur
   );
```

Version avec modificateur

```
bool  ObjectSetDouble(
   long                          chart_id,          // identificateur du graphique
   string                        name,              // nom
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // propriété
   int                           prop_modifier,     // modificateur
   double                        prop_value         // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être l'une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Le modificateur de la propriété indiquée. La plupart des propriétés ne nécessite pas de modificateur. Signifie le numéro du niveau pour les outils de Fibonacci et pour l'objet Fourchette d'Andrews. La numérotation des niveaux commence à 0.

prop_value

[in] La valeur de la propriété.

Valeur de Retour

Retourne true seulement si la commande de changement de la propriété de l'objet graphique est envoyée avec succès au graphique, sinon retourne false. Pour obtenir des informations supplémentaires sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

© 2000-2024, MetaQuotes Ltd.

La fonction utilise un appel asynchrone, ce qui signifie que la fonction n'attend pas la fin de l'exécution de la commande qui a été ajoutée dans la queue du graphique spécifié. Au lieu de cela, elle retourne le contrôle immédiatement.

Pour vérifier le résultat de l'exécution de la commande, vous pouvez utiliser une fonction demandant les propriétés de l'objet. Vous devriez cependant conserver à l'esprit que ce genre de fonctions est ajoutée à la fin de la queue de ce graphique, et qu'elles attendent le résultat de l'exécution, et peuvent donc être consommatrices en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

Exemple de création d'un objet de Fibonacci et d'ajout d'un nouveau niveau

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tableaux auxiliaires
   double high[],low[],price1,price2;
   datetime time[],time1,time2;
//--- copie les prix d'ouverture des 100 dernières barres
   int copied=CopyHigh(Symbol(),0,0,100,high);
   if(copied<=0)
     {
      Print("Echec de la copie des valeurs des prix High");
      return;
     }
//--- copie les prix de clôture des 100 dernières barres
   copied=CopyLow(Symbol(),0,0,100,low);
   if(copied<=0)
     {
      Print("Echec de la copie des valeurs des prix Low");
      return;
     }
//--- copie les heures d'ouvertures des 100 dernières barres
   copied=CopyTime(Symbol(),0,0,100,time);
   if(copied<=0)
     {
      Print("Echec de la copie des valeurs des heures d'ouverture");
      return;
     }
//--- inverse l'accès aux données copiées (ne sont plus des séries temporelles)
   ArraySetAsSeries(high,true);
   ArraySetAsSeries(low,true);
   ArraySetAsSeries(time,true);

//--- coordonnées du premier point d'ancrage de l'objet de Fibonacci
   price1=high[70];
   time1=time[70];
//--- coordonnées du premier point d'ancrage de l'objet de Fibonacci
```

© 2000-2024, MetaQuotes Ltd.

```
   price2=low[50];
   time2=time[50];

//--- il est temps de créer l'objet de Fibonacci
   bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
   if(created) // si l'objet est créé avec succès
     {
      //--- définition de la couleur des niveaux de Fibonacci
      ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
      //--- retourne le nombre de niveaux de Fibonacci
      int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
      Print("Niveaux de Fibonacci avant = ",levels);
      //--- écrit dans le journal le numéro du niveau et sa desription
      for(int i=0;i<levels;i++)
        {
         Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
               "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
        }
      //--- augmente le nombre de niveaux
      bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
      if(!modified) // échec du changement du nombre de niveaux
        {
         Print("Echec de changement du nombre de niveaux de Fibonacci, erreur ",GetLas
        }
      //--- nombre de niveaux après
      Print("Niveaux de Fibonacci après = ",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS))
      //--- spécifie la valeur pour le nouveau niveau créé
      bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
      if(added) // le niveau a été modifié
        {
         Print("Succès de l'ajout du niveau de Fibonacci");
         //--- définit la description du niveau
         ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
         ChartRedraw(0);
         //--- récupère la valeur actuelle du nombre de niveaux dans l'objet Fibonacci
         levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
         Print("Niveaux de Fibonacci après l'ajout = ",levels);
         //--- récupère tous les niveaux pour vérification
         for(int i=0;i<levels;i++)
           {
            Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                  "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
           }

        }
      else // échec de la tentative d'augmentation du nombre de niveaux dans l'objet F
        {
         Print("Echec de l'ajout d'un nouveau niveau de Fibonacci. L'erreur ",GetLastE
        }
```

© 2000-2024, MetaQuotes Ltd.

```
        }
    }
```

Voir aussi

Les types des objets, les propriétés des objets

© 2000-2024, MetaQuotes Ltd.

ObjectSetInteger

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type datetime, int, color, bool ou char. Il y a 2 variantes de la fonction.

Version sans modificateur

```
bool  ObjectSetInteger(
   long                              chart_id,          // identificateur du graphique
   string                            name,              // nom
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // propriété
   long                              prop_value         // valeur
   );
```

Version avec modificateur

```
bool  ObjectSetInteger(
   long                              chart_id,          // identificateur du graphique
   string                            name,              // nom
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // propriété
   int                               prop_modifier,     // modificateur
   long                              prop_value         // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être l'une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Le modificateur de la propriété indiquée. La plupart des propriétés ne nécessite pas de modificateur. Signifie le numéro du niveau pour les outils de Fibonacci et pour l'objet Fourchette d'Andrews. La numérotation des niveaux commence à 0.

prop_value

[in] La valeur de la propriété.

Valeur de Retour

Retourne true seulement si la commande de changement de la propriété de l'objet graphique est envoyée avec succès au graphique, sinon retourne false. Pour obtenir des informations supplémentaires sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

© 2000-2024, MetaQuotes Ltd.

La fonction utilise un appel asynchrone, ce qui signifie que la fonction n'attend pas la fin de l'exécution de la commande qui a été ajoutée dans la queue du graphique spécifié. Au lieu de cela, elle retourne le contrôle immédiatement.

Pour vérifier le résultat de l'exécution de la commande, vous pouvez utiliser une fonction demandant les propriétés de l'objet. Vous devriez cependant conserver à l'esprit que ce genre de fonctions est ajoutée à la fin de la queue de ce graphique, et qu'elles attendent le résultat de l'exécution, et peuvent donc être consommatrices en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

Exemple de création du tableau des couleurs Web

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                Table of Web Colors|
//|                          Copyright 2011, MetaQuotes Software Corp |
//|                                          https://www.metaquotes.net |
//+------------------------------------------------------------------+
#define X_SIZE 140      // la largeur de l'objet OBJ_EDIT
#define Y_SIZE 33       // la largeur de l'objet OBJ_EDIT
//+------------------------------------------------------------------+
//| Le tableau des couleurs Web                                      |
//+------------------------------------------------------------------+
color ExtClr[140]=
  {
   clrAliceBlue,clrAntiqueWhite,clrAqua,clrAquamarine,clrAzure,clrBeige,clrBisque,clrB
   clrBlue,clrBlueViolet,clrBrown,clrBurlyWood,clrCadetBlue,clrChartreuse,clrChocolate
   clrCornsilk,clrCrimson,clrCyan,clrDarkBlue,clrDarkCyan,clrDarkGoldenrod,clrDarkGray
   clrDarkMagenta,clrDarkOliveGreen,clrDarkOrange,clrDarkOrchid,clrDarkRed,clrDarkSalm
   clrDarkSlateBlue,clrDarkSlateGray,clrDarkTurquoise,clrDarkViolet,clrDeepPink,clrDee
   clrDodgerBlue,clrFireBrick,clrFloralWhite,clrForestGreen,clrFuchsia,clrGainsboro,cl
   clrGoldenrod,clrGray,clrGreen,clrGreenYellow,clrHoneydew,clrHotPink,clrIndianRed,cl
   clrLavender,clrLavenderBlush,clrLawnGreen,clrLemonChiffon,clrLightBlue,clrLightCora
   clrLightGoldenrod,clrLightGreen,clrLightGray,clrLightPink,clrLightSalmon,clrLightSe
   clrLightSlateGray,clrLightSteelBlue,clrLightYellow,clrLime,clrLimeGreen,clrLinen,cl
   clrMediumAquamarine,clrMediumBlue,clrMediumOrchid,clrMediumPurple,clrMediumSeaGreen
   clrMediumSpringGreen,clrMediumTurquoise,clrMediumVioletRed,clrMidnightBlue,clrMintC
   clrNavajoWhite,clrNavy,clrOldLace,clrOlive,clrOliveDrab,clrOrange,clrOrangeRed,clrO
   clrPaleGreen,clrPaleTurquoise,clrPaleVioletRed,clrPapayaWhip,clrPeachPuff,clrPeru,c
   clrPurple,clrRed,clrRosyBrown,clrRoyalBlue,clrSaddleBrown,clrSalmon,clrSandyBrown,c
   clrSienna,clrSilver,clrSkyBlue,clrSlateBlue,clrSlateGray,clrSnow,clrSpringGreen,clr
   clrThistle,clrTomato,clrTurquoise,clrViolet,clrWheat,clrWhite,clrWhiteSmoke,clrYell
  };
//+------------------------------------------------------------------+
//| La création et l'initialisation de l'objet OBJ_EDIT              |
//+------------------------------------------------------------------+
void CreateColorBox(int x,int y,color c)
  {
//--- générerons le nom pour un nouvel objet selon le nom de la couleur
   string name="ColorBox_"+(string)x+"_"+(string)y;
//--- créons un nouvel objet OBJ_EDIT
   if(!ObjectCreate(0,name,OBJ_EDIT,0,0,0))
     {
      Print("On n'a pas réussi à créer l'objet: '",name,"'");
      return;
     }
//--- spécifions les coordonnées du point d'ancrage, la largeur et la hauteur en pixel
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x*X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y*Y_SIZE);
   ObjectSetInteger(0,name,OBJPROP_XSIZE,X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YSIZE,Y_SIZE);
//---définissons la couleur du texte pour l'objet
   if(clrBlack==c) ObjectSetInteger(0,name,OBJPROP_COLOR,clrWhite);
   else            ObjectSetInteger(0,name,OBJPROP_COLOR,clrBlack);
//--- définissons la couleur du fond
   ObjectSetInteger(0,name,OBJPROP_BGCOLOR,c);
//--- définissons le texte de l'objet OBJ_EDIT correspondant à la couleur du fond
   ObjectSetString(0,name,OBJPROP_TEXT,(string)c);
  }
//+------------------------------------------------------------------+
//| La fonction du lancement du script à exécuter                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

```
//--- créons le tableau des blocs colorés 7x20
   for(uint i=0;i<140;i++)
      CreateColorBox(i%7,i/7,ExtClr[i]);
  }
```

Voir aussi

Les types des objets, les propriétés des objets

© 2000-2024, MetaQuotes Ltd.

ObjectSetString

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type string. Il y a 2 variantes de la fonction.

Version sans modificateur

```
bool  ObjectSetString(
   long                          chart_id,          // identificateur du graphique
   string                        name,              // nom
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // propriété
   string                        prop_value         // valeur
   );
```

Version avec modificateur

```
bool  ObjectSetString(
   long                          chart_id,          // identificateur du graphique
   string                        name,              // nom
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // propriété
   int                           prop_modifier,     // modificateur
   string                        prop_value         // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être l'une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Le modificateur de la propriété indiquée. La plupart des propriétés ne nécessite pas de modificateur. Signifie le numéro du niveau pour les outils de Fibonacci et pour l'objet Fourchette d'Andrews. La numérotation des niveaux commence à 0.

prop_value

[in] La valeur de la propriété.

Valeur de Retour

Retourne true seulement si la commande de changement de la propriété de l'objet graphique est envoyée avec succès au graphique, sinon retourne false. Pour obtenir des informations supplémentaires sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

© 2000-2024, MetaQuotes Ltd.

La fonction utilise un appel asynchrone, ce qui signifie que la fonction n'attend pas la fin de l'exécution de la commande qui a été ajoutée dans la queue du graphique spécifié. Au lieu de cela, elle retourne le contrôle immédiatement.

Pour vérifier le résultat de l'exécution de la commande, vous pouvez utiliser une fonction demandant les propriétés de l'objet. Vous devriez cependant conserver à l'esprit que ce genre de fonctions est ajoutée à la fin de la queue de ce graphique, et qu'elles attendent le résultat de l'exécution, et peuvent donc être consommatrices en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

Lors du changement de nom d'un objet graphique, deux événements sont générés, qui peuvent être traité dans les experts ou dans les indicateurs avec la fonction OnChartEvent() :

- l'événement de la suppression de l'objet avec le nom précédent ;
- l'événement de la création de l'objet graphique avec un nouveau nom.

© 2000-2024, MetaQuotes Ltd.

ObjectGetDouble

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type double. Il y a 2 variantes de la fonction.

1. Retourne immédiatement la valeur de la propriété.

```
double  ObjectGetDouble(
   long                          chart_id,          // identificateur du graphique
   string                        name,              // nom de l'objet
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // identificateur de la propriét
   int                           prop_modifier=0    // modificateur de la propriété
   );
```

2. Retourne true ou false en fonction du succès de l'exécution de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable de réception transmise par référence (le dernier paramètre).

```
bool  ObjectGetDouble(
   long                          chart_id,          // identificateur du graphique
   string                        name,              // nom de l'objet
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // identificateur de la propriét
   int                           prop_modifier,     // modificateur de la propriété
   double&                       double_var         // valeur de la propriété
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être l'une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Le modificateur de la propriété indiquée. Pour la première variante, par défaut la valeur du modificateur est égal à 0. La plupart des propriétés ne nécessite pas de modificateur. Signifie le numéro du niveau pour les outils de Fibonacci et pour l'objet Fourchette d'Andrews. La numérotation des niveaux commence à 0.

double_var

[out] La variable du type double, prenant la valeur de la propriété demandée.

Valeur de Retour

La valeur du type double pour la première variante de l'appel.

© 2000-2024, MetaQuotes Ltd.

Pour la deuxième variante de l'appel, retourne true si cette propriété est disponible et que la valeur a été copiée dans la variable double_var, sinon retourne false. Pour obtenir des informations supplémentaires sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

La fonction utilise un appel synchrone, ce qui signifie que la fonction attend la fin de l'exécution de toutes les commandes présentes dans la queue de ce graphique avant cet appel. C'est pourquoi cette fonction peut être consommatrice en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

© 2000-2024, MetaQuotes Ltd.

ObjectGetInteger

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type datetime, int, color, bool ou char. Il y a 2 variantes de la fonction.

1. Retourne immédiatement la valeur de la propriété.

```
long  ObjectGetInteger(
   long                              chart_id,          // identificateur du graphique
   string                            name,              // nom de l'objet
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // identificateur de la propri
   int                               prop_modifier=0    // modificateur de la propriété
   );
```

2. Retourne true ou false en fonction du succès de l'exécution de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable de réception transmise par référence (le dernier paramètre).

```
bool  ObjectGetInteger(
   long                              chart_id,          // identificateur du graphique
   string                            name,              // nom de l'objet
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // identificateur de la propri
   int                               prop_modifier,     // modificateur de la propriété
   long&                             long_var           // valeur de la propriété
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être l'une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Le modificateur de la propriété indiquée. Pour la première variante, par défaut la valeur du modificateur est égal à 0. La plupart des propriétés ne nécessite pas de modificateur. Signifie le numéro du niveau pour les outils de Fibonacci et pour l'objet Fourchette d'Andrews. La numérotation des niveaux commence à 0.

long_var

[out] La valeur du type long, prenant la valeur de la propriété demandée.

Valeur de Retour

La valeur du type long pour la première variante de l'appel.

© 2000-2024, MetaQuotes Ltd.

Pour la deuxième variante de l'appel, retourne true si cette propriété est disponible et que la valeur a été copiée dans la variable double_var, sinon retourne false. Pour obtenir des informations supplémentaires sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

La fonction utilise un appel synchrone, ce qui signifie que la fonction attend la fin de l'exécution de toutes les commandes présentes dans la queue de ce graphique avant cet appel. C'est pourquoi cette fonction peut être consommatrice en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

© 2000-2024, MetaQuotes Ltd.

ObjectGetString

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type string. Il y a 2 variantes de la fonction.

1. Retourne immédiatement la valeur de la propriété.

```
string  ObjectGetString(
   long                          chart_id,          // identificateur du graphique
   string                        name,              // nom de l'objet
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // identificateur de la propriét
   int                           prop_modifier=0    // modificateur de la propriété
   );
```

2. Retourne true ou false en fonction du succès de l'exécution de la fonction. En cas de succès, la valeur de la propriété est placée dans la variable de réception transmise par référence (le dernier paramètre).

```
bool  ObjectGetString(
   long                          chart_id,          // identificateur du graphique
   string                        name,              // nom de l'objet
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // identificateur de la propriét
   int                           prop_modifier,     // modificateur de la propriété
   string&                       string_var         // prendra la valeur de la propr
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être l'une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Le modificateur de la propriété indiquée. Pour la première variante, par défaut la valeur du modificateur est égal à 0. La plupart des propriétés ne nécessite pas de modificateur. Signifie le numéro du niveau pour les outils de Fibonacci et pour l'objet Fourchette d'Andrews. La numérotation des niveaux commence à 0.

string_var

[out] La valeur du type string, prenant la valeur de la propriété demandée.

Valeur de Retour

La valeur du type string pour la première variante de l'appel.

© 2000-2024, MetaQuotes Ltd.

Pour la deuxième variante de l'appel, retourne true si cette propriété est disponible et que la valeur a été copiée dans la variable double_var, sinon retourne false. Pour obtenir des informations supplémentaires sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

La fonction utilise un appel synchrone, ce qui signifie que la fonction attend la fin de l'exécution de toutes les commandes présentes dans la queue de ce graphique avant cet appel. C'est pourquoi cette fonction peut être consommatrice en terme de temps. Cette caractéristique doit être prise en compte lors de l'utilisation d'un grand nombre d'objets sur un graphique.

Lors du changement de nom d'un objet graphique, deux événements sont générés, qui peuvent être traité dans les experts ou dans les indicateurs avec la fonction OnChartEvent() :

- l'événement de la suppression de l'objet avec le nom précédent ;
- l'événement de la création de l'objet graphique avec un nouveau nom.

© 2000-2024, MetaQuotes Ltd.

TextSetFont

Établit la fonte pour la sortie du texte par les méthodes du dessin et rend le résultat de la réussite de cette opération. On utilise par défaut la fonte Arial et la taille -120 (12 pt).

```
bool  TextSetFont(
   const string  name,            // le nom de la fonte ou la voie vers le fichier de
   int           size,            // la taille de la fonte
   uint          flags,           // la combinaison des drapeaux
   int           orientation=0    // l'angle de l'inclinaison du texte
   );
```

Réglages

name

[in] Le nom de la fonte dans le système, ou le nom de la ressource contenant la fonte, ou la voie vers le fichier de la fonte sur le disque.

size

[in] La taille de la fonte, qui peut être spécifiée par les valeurs positives et négatives. Aux valeurs positives la taille du texte déduit ne dépend pas des réglages des tailles des fontes dans le système d'exploitation. Aux valeurs négatives la valeur est spécifiée dans les dixièmes parts du point et la taille du texte dépendra des réglages du système ("l'échelle standard" ou "une grande échelle"). Plus en détail sur la différence entre les modes regardez dans la Note

flags

[in] La combinaison des drapeaux, décrivant le style de la fonte.

orientation

[in] L'angle de l'inclinaison du texte à l'horizontale vers l'axe X, l'unité de mesure est égale 0.1 degrés. C'est-à-dire `orientation=450`signifie l'inclinaison à 45 degrés.

La valeur rendue

Rend true en cas de l'installation réussie de la fonte courante, autrement rend false. Les codes possibles des erreurs:

- ERR_INVALID_PARAMETER(4003) - *name* présente NULL ou "" (la ligne vide),
- ERR_INTERNAL_ERROR(4001) - l'erreur du système d'exploitation (par exemple, la tentative de la création de la fonte inexistante).

Note

Si dans le nom de la fonte est utilisé "::", la fonte est chargée de la ressource EX5. Si le nom de la fonte*name*est indiqué avec l'extension, la fonte est chargée du fichier, en cela - si la voie commence par "\" ou "/", le fichier est cherché par rapport au catalogue MQL5, autrement il est cherché par rapport à la voie du fichier EX5 qui a appelé la fonction TextSetFont().

La taille de la fonte est spécifiée par les valeurs positives ou négatives, le signe définit la dépendance de la taille du texte des réglages du système d'exploitation (l'échelle de la fonte).

- Si le taille est spécifiée par le nombre positif, à l'affichage de la fonte logique au physique se passe la transformation de la taille en unités physiques de la mesure du dispositif (les pixels) et cette taille correspond à la hauteur des cellules des symboles des fontes accessibles. Il n'est pas recommandé dans les cas où on suppose l'utilisation partagée sur le graphique des textes déduits

© 2000-2024, MetaQuotes Ltd.

par la fonctionTextOut(), et des textes affichés à l'aide de l'objet graphiqueOBJ_LABEL ("La marque de texte").

- Si la taille est spécifiée par le nombre négatif, la taille indiquée est supposé spécifiée aux dixièmes parts du point logique (la valeur -350 est égal au 35 points logiques) et se divise sur 10, et puis la valeur reçue sera transformée en unités physiques de la mesure du dispositif (les pixels) et correspond à la valeur absolue de la hauteur du symbole des fontes accessibles. Pour recevoir à l'écran le texte de la même taille, comme dans l'objetOBJ_LABEL , prenez la taille de la fonte indiquée aux propriétés de l'objet et multipliez sur -10.

Les drapeaux peuvent être utilisés en forme de la combinaison des drapeaux du style avec un des drapeaux, qui spécifie l'épaisseur de la fonte. Les noms des drapeaux sont plus bas.

Les drapeaux pour spécifier le style du tracé de la fonte

Le drapeau	La description
FONT_ITALIC	L'italique
FONT_UNDERLINE	Le soulignement
FONT_STRIKEOUT	Le barrement

Les drapeaux pour spécifier l'épaisseur de la fonte

Le drapeau
FW_DONTCARE
FW_THIN
FW_EXTRALIGHT
FW_ULTRALIGHT
FW_LIGHT
FW_NORMAL
FW_REGULAR
FW_MEDIUM
FW_SEMIBOLD
FW_DEMIBOLD
FW_BOLD
FW_EXTRABOLD
FW_ULTRABOLD
FW_HEAVY
FW_BLACK

Voir aussi

Les ressources , ResourceCreate(), ResourceSave(), TextOut()

© 2000-2024, MetaQuotes Ltd.

TextOut

Déduit le texte au tableau d'utilisateur (le tampon) et rend le résultat de la réussite de cette opération. Ce tableau est conçu pour créer le ressourcegraphique.

```
bool  TextOut(
   const string       text,          // le texte déduit
   int                x,             // la coordonnée X
   int                y,             // la coordonnée Y
   uint               anchor,        // le moyen du rattachement
   uint               &data[],       // le tampon pour la sortie
   uint               width,         // la largeur du tampon en points
   uint               height,        // la hauteur du tampon en points
   uint               color,         // la couleur du texte
   ENUM_COLOR_FORMAT  color_format   // le format de la couleur pour la sortie
   );
```

Réglages

text

[in] Le texte déduit, qui sera enregistré au tampon. Seulement une sortie de texte d'une ligne est effectuée.

x

[in] La coordonnée X du point du rattachement du texte.

y

[in] La coordonnée Y du point du rattachement du texte.

anchor

[in] La valeur de l'ensemble de 9 moyens prédéterminés de la disposition du point du rattachement du texte déduit. Est spécifiée par la combinaison de deux drapeaux - le drapeau de l'alignement du texte à l'horizontale et le drapeau de l'alignement du texte selon la verticale. Les noms des drapeaux sont amenés dans la Note.

data[]

[in] Le tampon, où on déduit le texte. Ce tableau est utilisé pour créer le ressource graphique.

width

[in] La largeur du tampon en points (en pixels).

height

[in] La hauteur du tampon en points (en pixels).

color

[in] La couleur du texte.

color_format

[in] Le format de la couleur est spécifié par la valeur de l'énumérationENUM_COLOR_FORMAT.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement rend false.

© 2000-2024, MetaQuotes Ltd.

Note

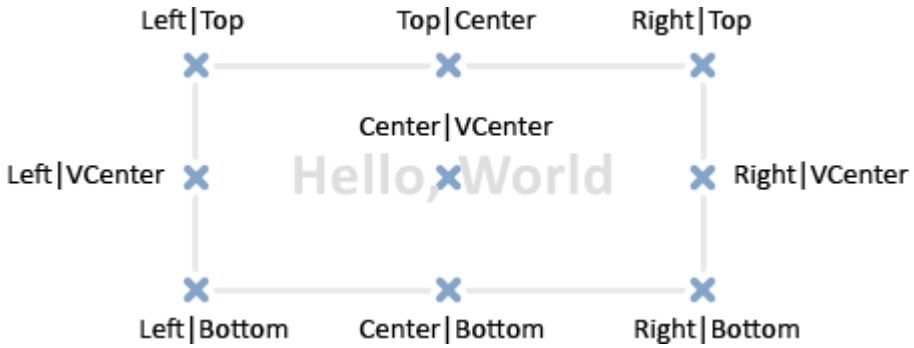
Le moyen du rattachement spécifié par le paramètre `anchor`, qui est la combinaison de deux drapeaux de l'alignement du texte selon la verticale et l'horizontale. Les drapeaux de l'alignement du texte à l'horizontale:

- TA_LEFT - le point du rattachement sur le côté gauche du rectangle limitant
- TA_CENTER - le point du rattachement à l'horizontale se trouve au milieu du rectangle limitant
- TA_RIGHT - le point du rattachement sur le côté droite du rectangle limitant

Les drapeaux de l'alignement du texte selon la verticale:

- TA_TOP - le point du rattachement sur le côté supérieur du rectangle limitant
- TA_VCENTER - le point du rattachement à la verticale se trouve au milieu du rectangle limitant
- TA_BOTTOM - le point du rattachement sur le côté inférieur du rectangle limitant

Les combinaisons possibles des drapeaux et les moyens du rattachement spécifiés par eux sont montrées sur le dessin.



Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//--- la largeur et la hauteur du canvas (la toile, où on fait le dessin)
#define IMG_WIDTH  200
#define IMG_HEIGHT 200
//--- avant le lancement du script montrons la fenêtre avec les paramètres
#property script_show_inputs
//---donnons la possibilité de spécifier le format de la couleur
input ENUM_COLOR_FORMAT clr_format=COLOR_FORMAT_XRGB_NOALPHA;
//--- le tableau (le tampon) pour la rendu
uint ExtImg[IMG_WIDTH*IMG_HEIGHT];
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- créeron l'objet OBJ_BITMAP_LABEL pour le dessin
   ObjectCreate(0,"CLOCK",OBJ_BITMAP_LABEL,0,0,0);
//--- indiquerons le nom de la ressource graphique pour la rendu dans l'objet CLOCK
   ObjectSetString(0,"CLOCK",OBJPROP_BMPFILE,"::IMG");

//--- les variables auxiliaires
   double a;            // l'angle de la flèche
   uint   nm=2700;      // le compteur des minutes
   uint   nh=2700*12;   // le compteur des heures
   uint   w,h;          // les variables pour la réception des tailles des lignes de t
   int    x,y;          // les variables pour le calcul des coordonnées courants du p

//--- tournons les aiguilles des heures dans le cycle infini, jusqu'à ce que le script
   while(!IsStopped())
     {
      //--- le nettoyage du tableau du tampon du dessin des heures
      ArrayFill(ExtImg,0,IMG_WIDTH*IMG_HEIGHT,0);
      //--- l'installation de la fonte pour la rendu des chiffres sur le cadran
      TextSetFont("Arial",-200,FW_EXTRABOLD,0);
      //--- dessinons le cadran
      for(int i=1;i<=12;i++)
        {
         //--- recevrons les tailles de l'heure courante sur le cadran
         TextGetSize(string(i),w,h);
         //--- calculons les coordonnées de l'heure courante sur le cadran
         a=-((i*300)%3600*M_PI)/1800.0;
         x=IMG_WIDTH/2-int(sin(a)*80+0.5+w/2);
         y=IMG_HEIGHT/2-int(cos(a)*80+0.5+h/2);
         //--- la sortie de cette heure sur le cadran au tampon ExtImg[]
         TextOut(string(i),x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,c
        }
      //---maintenant établissons la fonte pour la rendu de la grande aiguille
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nm%3600));
      //--- recevrons les tailles de la grande aiguille
      TextGetSize("----->",w,h);
      //--- calculons les coordonnées de la grande aiguille sur le cadran
      a=-(nm%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- la sortie de la grande aiguille sur le cadran au tampon ExtImg[]
      TextOut("----->",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_f

      //---maintenant établissons la fonte pour la rendu de la petite aiguille
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nh/12%3600));
      TextGetSize("==>",w,h);
      //--- calculons les coordonnées de la petite aiguille sur le cadran
      a=-(nh/12%3600*M_PI)/1800.0;
```

© 2000-2024, MetaQuotes Ltd.

```
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- la sortie de la petite aiguille sur le cadran au tampon ExtImg[]
      TextOut("==>",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_forr

      //--- la mise à jour de la ressource graphique
      ResourceCreate("::IMG",ExtImg,IMG_WIDTH,IMG_HEIGHT,0,0,IMG_WIDTH,clr_format);
      //---la mise à jour obligatoire du graphique
      ChartRedraw();

      //---augmentons les compteurs de l'heure et des minutes
      nm+=60;
      nh+=60;
      //--- tenons une courte pause entre les cadres
      Sleep(10);
     }
//--- supprimons l'objet CLOCK à l'achèvement du travail du script
   ObjectDelete(0,"CLOCK");
//---
  }
```

Voir aussi

Les ressources, ResourceCreate(), ResourceSave(), TextGetSize(), TextSetFont()

© 2000-2024, MetaQuotes Ltd.

TextGetSize

Rend la largeur et la hauteur de la ligneaux réglages courants de la fonte.

```
bool  TextGetSize(
   const string       text,          // la ligne du texte
   uint&               width,        // la largeur du tampon en points
   uint&               height        // la hauteur du tampon en points
   );
```

Réglages

text

[in] La ligne pour laquelle on reçoit la longueur et la largeur.

width

[out] Le paramètre d'entrée pour la réception de la largeur.

height

[out] Le paramètre d'entrée pour la réception de la hauteur.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement rend false. Les codes possibles des erreurs:

- ERR_INTERNAL_ERROR(4001) - en cas de l'erreur du système d'exploitation.

Voir aussi

Les ressources, ResourceCreate(), ResourceSave(), TextSetFont(), TextOut()

© 2000-2024, MetaQuotes Ltd.

Les fonctions pour le travail avec les indicateurs techniques

Toutes les fonctions du type iMA, iAC, iMACD, iIchimoku etc., créent une copie de l'indicateur technique correspondant dans la cache globale du terminal de client. Si la copie de l'indicateur avec ces paramètres existe déjà, une nouvelle copie n'est pas créée, mais augmente le compteur des liens à la copie donnée.

Ces fonctions rendent le handle de la copie correspondante de l'indicateur. En utilisant ce handle on peut recevoir par la suite les données calculées par l'indicateur correspondant. On peut copier les données du tampon correspondant (les indicateurs techniques contiennent les données calculées dans les tampons intérieurs, que, en fonction de l'indicateur, peut être de 1 jusqu'à 5) au programme mql5 à l'aide de la fonction CopyBuffer().

On ne peut pas s'adresser aux données de l'indicateur à la fois après sa création, puisque on demande un certain temps sur le calcul des valeurs de l'indicateur, et c'est pour cela c'est mieux de créer les handles des indicateurs à OnInit(). La fonction iCustom() crée l'indicateur correspondant d'utilisateur et à la création réussie rend son handle. Les indicateurs d'utilisateur peuvent contenir jusqu'à 512 des tampons d'indicateur, dont le contenu peut être reçu à l'aide de la fonction CopyBuffer(), en utilisant le handle reçu.

Il y a une méthode universelle de la création de n'importe quel indicateur technique avec l'aide de la fonction IndicatorCreate(). Cette fonction reçoit à titre des paramètres d'entrée:

- le nom du symbole;
- le temps trame;
- le type de l'indicateur créé;
- le nombre de paramètres d'entrée de l'indicateur;
- le tableau du type MqlParam, contenant tous les paramètres nécessaires d'entrée.

La mémoire informatique peut être libérée d'un indicateur qui n'est plus utilisé, en utilisant la fonction IndicatorRelease(), à laquelle le handle d'indicateur est transmise.

Note. L'appel multiple à la fonction de l'indicateur avec les mêmes paramètres dans la limite d'un programme mql5 n'amène pas à l'augmentation multiple du compteur des liens, le compteur sera augmenté seulement une fois sur 1. Il est recommandé de recevoir cependant les handles des indicateurs en fonction OnInit() ou dans le constructeur de la classe, avec l'utilisation ultérieure des handles reçus dans les autres fonctions. Les compteurs des liens diminue à la déinitialisation du programme mql5.

Toutes les fonctions d'indicateur ont au minimum 2 paramètres - le symbole et la période. La valeur du symbole NULL signifie l'instrument courant, la valeur de la période 0 signifie le temps trame courant.

Fonction	Rend le handle de l'indicateur:
iAC	Accelerator Oscillator
iAD	Accumulation/Distribution
iADX	Average Directional Index
iADXWilder	Average Directional Index by Welles Wilder
iAlligator	Alligator

© 2000-2024, MetaQuotes Ltd.

Fonction	Rend le handle de l'indicateur:
iAMA	Adaptive Moving Average
iAO	Awesome Oscillator
iATR	Average True Range
iBearsPower	Bears Power
iBands	Bollinger Bands®
iBullsPower	Bulls Power
iCCI	Commodity Channel Index
iChaikin	Chaikin Oscillator
iCustom	l'indicateur d'utilisateur
iDEMA	Double Exponential Moving Average
iDeMarker	DeMarker
iEnvelopes	Envelopes
iForce	Force Index
iFractals	Fractals
iFrAMA	Fractal Adaptive Moving Average
iGatore	Gator Oscillator
iIchimoku	Ichimoku Kinko Hyo
iBWMFI	Market Facilitation Index by Bill Williams
iMomentum	Momentum
iMFI	Money Flow Index
iMA	Moving Average
iOsMA	Moving Average of Oscillator (MACD histogram)
iMACD	Moving Averages Convergence-Divergence
iOBV	On Balance Volume
iSAR	Parabolic Stop And Reverse System
iRSI	Relative Strength Index
iRVI	Relative Vigor Index
iStdDev	Standard Deviation
iStochastic	Stochastic Oscillator
iTEMA	Triple Exponential Moving Average

© 2000-2024, MetaQuotes Ltd.

Fonction	Rend le handle de l'indicateur:
iTriX	Triple Exponential Moving Averages Oscillator
iWPR	Williams' Percent Range
iVIDyA	Variable Index Dynamic Average
iVolumes	Volumes

© 2000-2024, MetaQuotes Ltd.

iAC

Crée au cash global du terminal de client l'indicateur Accelerator Oscillator et rend son handle. Seulement un tampon.

```
int  iAC(
   string           symbol,      // nom du symbole
   ENUM_TIMEFRAMES  period       // période
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeur de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAC.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iAC."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- La construction iAC
#property indicator_label1  "iAC"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAC,                // utiliser iAC
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation              type=Call_iAC;          // le type de la fonction
input string                symbol=" ";             // le symbole
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // le temps trame
//--- les tampons d'indicateur
double iACBuffer[];
double iACColors[];
//--- la variable pour le stockage du handle de l'indicateur iAC
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockons le nombre de valeurs dans l'indicateur Accelerator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iAC)
      handle=iAC(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AC);
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- informons sur l'échec et déduisons le numéro d'erreur
       PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iAC pour une pa
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- le travail de l'indicateur est terminé avant terme
       return(INIT_FAILED);
      }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Accelerator Oscil
   short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iAC
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est le premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iACBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

```
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux iACBuffer et iACColors par les valeurs de l'indicateur
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Accelerator Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iAC         |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // le tampon d'indicateur des valeu
                          double &color_indexes[], // le tampon coloré (pour la conser
                          int ind_handle,          // le handle de l'indicateur iAC
                          int amount               // le nombre de valeurs copiées
                          )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iACBuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAC, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- maintenant nous copions les index des couleurs
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les valeurs des couleurs de l'indicateur
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iAD

Rend le handle de l'indicateur Accumulation/Distribution. Seulement un tampon.

```
int  iAD(
   string                symbol,              // nom du symbole
   ENUM_TIMEFRAMES       period,              // période
   ENUM_APPLIED_VOLUME   applied_volume       // type du volume pour le calcul
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

applied_volume

[in] Le volume utilisé. Peut être l'un des ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAD.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iAD."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iAD
#property indicator_label1  "iAD"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAD,                // utiliser iAD
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iAD;          // le type de la fonction
input ENUM_APPLIED_VOLUME  volumes;                // le volume utilisé
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- le tampon d'indicateur
double         iADBuffer[];
//--- la variable pour stocker le handle de l'indicateur iAD
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//---stockerons le nombre de valeurs dans l'indicateur Accumulation/Distribution
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iAD)
      handle=iAD(name,period,volumes);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=volumes;
      handle=IndicatorCreate(name,period,IND_AD,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iAD pour la pai
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Accu
   short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iAD
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

```
      //--- si le tableau iADBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iADBuffer par les valeurs de l'indicateur Accumulation/Di
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Accumulation/Distribution
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iAD         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // le tampon d'indicateur de la ligne Acc
                         int ind_handle,     // le handle de l'indicateur iAD
                         int amount          // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iADBuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAD, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iADX

Rend le handle de l'indicateur Average Directional Movement Index.

```
int  iADX(
   string           symbol,         // nom du symbole
   ENUM_TIMEFRAMES  period,         // période
   int              adx_period      // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

adx_period

[in] La période pour le calcul de l'index.

Note

Les numéros des tampons: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iADX.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iADX."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

```
//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADX,              // utiliser iADX
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iADX;         // le type de la fonction
input int                  adx_period=14;          // la période du calcul
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- les tampons d'indicateur
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- la variable pour le stockage du handle de l'indicateur iADX
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Average Directional Movement I
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iADX)
      handle=iADX(name,period,adx_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADX,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iADX pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Aver
   short_name=StringFormat("iADX(%s/%s period=%d)",name,EnumToString(period),adx_perio
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iADX
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau ADXBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Average Directional Mover
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur Average Directional Movement I
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iADX        |
//+------------------------------------------------------------------+
```

```
bool FillArraysFromBuffers(double &adx_values[],      // le tampon d'indicateur de la
                           double &DIplus_values[],   // le tampon d'indicateur pour I
                           double &DIminus_values[],  // le tampon d'indicateur pour I
                           int ind_handle,            // le handle de l'indicateur iAI
                           int amount                 // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau ADXBuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADX, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau DI_plusBuffer par les valeurs du tampon d'indic
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADX, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau DI_plusBuffer par les valeurs du tampon d'indic
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADX, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iADXWilder

Rend le handle de l'indicateur Average Directional Movement Index by Welles Wilder.

```
int  iADXWilder(
   string           symbol,         // nom du symbole
   ENUM_TIMEFRAMES  period,         // période
   int              adx_period      // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

adx_period

[in] La période pour le calcul de l'index.

Note

Les numéros des tampons: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   iADXWilder.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iADXWilder
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

```
//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADXWilder,         // utiliser iADXWilder
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iADXWilder;   // le type de la fonction
input int                  adx_period=14;          // la période du calcul
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- les tampons d'indicateur
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- la variable pour stocker le handle de l'indicateur iADXWilder
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Average Directional Movement I
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iADXWilder)
      handle=iADXWilder(name,period,adx_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADXW,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iADXWilder pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Aver
   short_name=StringFormat("iADXWilder(%s/%s period=%d)",name,EnumToString(period),adx
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2024, MetaQuotes Ltd.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iADXWilder
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau ADXBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Average Directional Mover
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur Average Directional Movement I
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iADXWilder  |
//+------------------------------------------------------------------+
```

```
bool FillArraysFromBuffers(double &adx_values[],      // le tampon d'indicateur de la
                           double &DIplus_values[],   // le tampon d'indicateur pour I
                           double &DIminus_values[],  // le tampon d'indicateur pour I
                           int ind_handle,            // le handle de l'indicateur iAI
                           int amount                 // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau ADXBuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADXWilder,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau DI_plusBuffer par les valeurs du tampon d'indic
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADXWilder,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau DI_plusBuffer par les valeurs du tampon d'indic
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADXWilder,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iAlligator

Rend le handle de l'indicateur Alligator.

```
int  iAlligator(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 jaw_period,        // période pour le calcul des mâchoires
   int                 jaw_shift,         // décalage horizontal des mâchoires
   int                 teeth_period,      // période pour le calcul des dents
   int                 teeth_shift,       // décalage horizontal des dents
   int                 lips_period,       // période pour le calcul des lèvres
   int                 lips_shift,        // décalage horizontal des lèvres
   ENUM_MA_METHOD      ma_method,         // type de lissage
   ENUM_APPLIED_PRICE  applied_price      // type de prix ou handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

jaw_period

[in] La période de la prise de moyenne de la ligne bleue (la mâchoire d'alligator).

jaw_shift

[in] Le décalage de la ligne bleue par rapport au graphique du prix.

teeth_period

[in] La période de la prise de moyenne de la ligne rouge (les dents d'alligator).

teeth_shift

[in] Le déplacement de la ligne rouge par rapport au graphique du prix.

lips_period

[in] La période de la prise de moyenne de la ligne verte (les lèvres d'alligator).

lips_shift

[in] Le déplacement de la ligne verte par rapport au graphique du prix.

ma_method

[in] La méthode de la prise de moyenne.Peut être chacune des valeurs de l'énumération ENUM_MA_METHOD.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

© 2000-2024, MetaQuotes Ltd.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - GATORJAW_LINE, 1 - GATORTEETH_LINE, 2 - GATORLIPS_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                              Demo_iAlligator.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iAlligator
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans l'Alligator standard."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- plot Jaws
#property indicator_label1  "Jaws"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot Teeth
#property indicator_label2  "Teeth"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot Lips
#property indicator_label3  "Lips"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
```

```
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAlligator,        // utiliser iAlligator
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iAlligator;   // le type de la fonction
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
input int                  jaw_period=13;          // la période pour la ligne des Mâc
input int                  jaw_shift=8;            // le décalage de la ligne des Mâch
input int                  teeth_period=8;         // la période pour la ligne des Der
input int                  teeth_shift=5;          // le décalage de la ligne des Mâch
input int                  lips_period=5;          // la période pour la ligne des Lèv
input int                  lips_shift=3;           // le décalage de la ligne des Lèvr
input ENUM_MA_METHOD       MA_method=MODE_SMMA;    // la méthode de la prise de moyenn
input ENUM_APPLIED_PRICE   applied_price=PRICE_MEDIAN;// le type de prix utilisé pour
//--- les tampons d'indicateur
double         JawsBuffer[];
double         TeethBuffer[];
double         LipsBuffer[];
//--- la variable pour stocker le handle de l'indicateur iAlligator
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Alligator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
//--- spécifions le décalage pour chaque ligne
   PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
```

© 2000-2024, MetaQuotes Ltd.

```
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iAlligator)
      handle=iAlligator(name,period,jaw_period,jaw_shift,teeth_period,
                        teeth_shift,lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[8];
     //--- les périodes et les décalages des lignes de l'Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
//---le type du lissage
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
//--- le type du prix
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
//--- créons le handle
      handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iAlligator pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Alli
   short_name=StringFormat("iAlligator(%s/%s, %d,%d,%d,%d,%d,%d)",name,EnumToString(pe
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,
```

© 2000-2024, MetaQuotes Ltd.

```
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iAlligator
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau JawsBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur Alligator
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,l
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
```

```
                                values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Alligator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iAlligator  |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &jaws_buffer[],  // le tampon d'indicateur pour la l
                           int j_shift,            // le décalage de la ligne des Mâch
                           double &teeth_buffer[], // le tampon d'indicateur pour la l
                           int t_shift,            // le décalage de la ligne des Dent
                           double &lips_buffer[],  // le tampon d'indicateur pour la l
                           int l_shift,            // le décalage de la ligne des Lèvr
                           int ind_handle,         // le handle de l'indicateur iAllig
                           int amount              // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau JawsBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAlligator,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau TeethBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAlligator,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau LipsBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAlligator,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iAMA

Rend le handle de l'indicateur Adaptive Moving Average. Seulement un tampon.

```
int  iAMA(
   string              symbol,             // nom du symbole
   ENUM_TIMEFRAMES     period,             // période
   int                 ama_period,         // période AMA
   int                 fast_ma_period,     // période de la glissante rapide
   int                 slow_ma_period,     // période de la glissante lente
   int                 ama_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_APPLIED_PRICE  applied_price       // type du prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ama_period

[in] La période du calcul, dont on calcule le coefficient d'efficacité.

fast_ma_period

[in] La période rapide pour le calcul du coefficient de l'effacement aux moments du marché rapide.

slow_ma_period

[in] La période lente pour le calcul du coefficient de l'effacement dans l'absence de la tendance.

ama_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iAMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iAMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans AMA standard."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iAMA
#property indicator_label1  "iAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAMA,              // utiliser iAMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iAMA;          // le type de la fonction
input string               symbol=" ";              // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;   // le temps trame
input int                  ama_period=15;           // la période pour le calcul
input int                  fast_ma_period=2;        // la période de la glissante rapi
input int                  slow_ma_period=30;       // la période de la glissante lent
input int                  ama_shift=0;             // le décalage par l'horizontale
input ENUM_APPLIED_PRICE   applied_price;           // le type du prix
//--- le tampon d'indicateur
double         iAMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iAMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Adaptive Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
```

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
//--- spécifions le décalage
   PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iAMA)
      handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,appli
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[5];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ama_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=fast_ma_period;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slow_ma_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ama_shift;
      //--- le type du prix
      pars[4].type=TYPE_INT;
      pars[4].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_AMA,5,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iAMA pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Adaptive Moving Av
   short_name=StringFormat("iAMA(%s/%s,%d,%d,%d,d)",name,EnumToString(period),ama_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iAlligator
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iADBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur Alligator
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
```

© 2000-2024, MetaQuotes Ltd.

```
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Accelerator Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iAMA          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // le tampon d'indicateur de la ligne
                         int a_shift,           // le décalage de la ligne AMA
                         int ind_handle,        // le handle de l'indicateur iAMA
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iAMABuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAlligator,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iAO

Rend le handle de l'indicateur Awesome oscillator. Seulement un tampon.

```
int  iAO(
   string           symbol,      // nom du symbole
   ENUM_TIMEFRAMES  period       // période
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAO.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iAO."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- la construction iAO
#property indicator_label1  "iAO"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen,clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
```

```
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAO,                // utiliser iAO
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iAO;          // le type de la fonction
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- les tampons d'indicateur
double         iAOBuffer[];
double         iAOColors[];
//--- la variable pour stocker le handle de l'indicateur iAO
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Awesome Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iAOColors,INDICATOR_COLOR_INDEX);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iAO)
      handle=iAO(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AO);
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- informons sur l'échec et déduisons le numéro d'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iAO pour la pai
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Awesome Oscillato
   short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iAO
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iAOBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
```

```
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux iAOBuffer et iAOColors par les valeurs de l'indicateur
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffer(iAOBuffer,iAOColors,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Accelerator Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iAO         |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // le tampon d'indicateur des valeu
                          double &color_indexes[], // le tampon coloré (pour la conser
                          int ind_handle,          // le handle de l'indicateur iAO
                          int amount               // le nombre de valeurs copiées
                          )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iAOBuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAO, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- maintenant nous copions les index des couleurs
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les valeurs des couleurs de l'indicateur
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iATR

Rend le handle de l'indicateur Average True Range. Seulement un tampon.

```
int  iATR(
   string           symbol,        // nom du symbole
   ENUM_TIMEFRAMES  period,        // période
   int              ma_period      // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iATR.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iATR."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iATR
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iATR,// utiliser iATR
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input int                  atr_period=14;          // la période pour le calcul
input Creation             type=Call_iATR;          // le type de la fonction
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- le tampon d'indicateur
double         iATRBuffer[];
//--- la variable pour le stockage du handle de l'indicateur iAC
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Average True Range
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iATRBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iATR)
      handle=iATR(name,period,atr_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=atr_period;
      handle=IndicatorCreate(name,period,IND_ATR,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro d'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iATR pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Ave
   short_name=StringFormat("iATR(%s/%s, period=%d)",name,EnumToString(period),atr_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iATR
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

```
      //--- si le tableau iATRBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iATRBuffer  par les valeurs de l'indicateur Average True
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Accelerator Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iATR          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs ATR
                         int ind_handle,    // le handle de l'indicateur iATR
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iATRBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iATR, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iBearsPower

Rend le handle de l'indicateur Bears Power. Seulement un tampon.

```
int  iBearsPower(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 ma_period,         // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBearsPower.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iBearsPowe
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iBearsPower
#property indicator_label1  "iBearsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBearsPower,       // utiliser iBearsPower
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iBearsPower;  // le type de la fonction
input int                  ma_period=13;           // la période de la glissante
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- le tampon d'indicateur
double         iBearsPowerBuffer[];
//--- a variable pour stocker le handle de l'indicateur iBearsPower
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Bears Power
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iBearsPower)
      handle=iBearsPower(name,period,ma_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- la période de la glissante
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro d'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iBearsPower pou
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Bears Powera été c
   short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iBearsPower
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- si le tableau iATRBuffer est plus grand, que les valeurs dans l'indicateu
       //--- autrement copions moins que la taille des tampons d'indicateur
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- signifie que notre indicateur est calculé non pour la première fois et dès
       //--- pour le calcul a été ajouté pas plus d'une barre
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- remplissons le tableau iBearsPowerBuffer par les valeurs de l'indicateur Bears P
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
    if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
    string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- déduisons le message de service sur le graphique
    Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur Bears Power
    bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
    return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iBearsPower   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Bear
                         int ind_handle,    // le handle de l'indicateur iBearsPower
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
    ResetLastError();
//--- remplissons la partie du tableau iBearsPowerBuffer par les valeurs du tampon d'i
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
      {
       //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
       PrintFormat("On n'a pas réussi à copier les données de l'indicateur iBearsPower,
       //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera con
       return(false);
      }
//--- tout a réussi
    return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iBands

Rend le handle de l'indicateur Bollinger Bands®.

```
int  iBands(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 bands_period,      // période pour le calcul de la moyenne ligr
   int                 bands_shift,       // décalage de l'indicateur à l'horizontale
   double              deviation,         // nombre de divergences standards
   ENUM_APPLIED_PRICE  applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

bands_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

bands_shift

[in] Le décalage l'indicateur par rapport au graphique des prix.

deviation

[in] La divergence de la ligne principale.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - BASE_LINE, 1 - UPPER_BAND, 2 - LOWER_BAND

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iBands.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
```

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iBands."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- la construction Upper
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Lower
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrMediumSeaGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- la construction Middle
#property indicator_label3  "Middle"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrMediumSeaGreen
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBands,            //utiliser iBands
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iBands;          // le type de la fonction
input int                  bands_period=20;           // la période de la glissante mo
input int                  bands_shift=0;             // le décalage
input double               deviation=2.0;             // le nombre de divergences star
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- les tampons d'indicateur
double         UpperBuffer[];
double         LowerBuffer[];
```

```
double         MiddleBuffer[];
//--- la variable pour stocker le handle de l'indicateur iBands
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Bollinger Bands
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- spécifions le décalage pour chaque ligne
   PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iBands)
      handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période de la glissante
      pars[0].type=TYPE_INT;
      pars[0].integer_value=bands_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=bands_shift;
      //--- le nombre de divergences standards
      pars[2].type=TYPE_DOUBLE;
      pars[2].double_value=deviation;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iBands pour la
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Bollinger Bands a
   short_name=StringFormat("iBands(%s/%s, %d,%d,%G,%s)",name,EnumToString(period),
                           bands_period,bands_shift,deviation,EnumToString(applied_pri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iBands
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- si la taille des tableaux d'indicateur est plus grand que des valeurs dans
       //--- autrement copions moins que la taille des tampons d'indicateur
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- signifie que notre indicateur est calculé non pour la première fois et dès
       //--- pour le calcul a été ajouté pas plus d'une barre
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- remplissons le tableau par les valeurs de l'indicateur Bollinger Bands
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
    if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle,v
//--- formerons le message
    string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- déduisons le message de service sur le graphique
    Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Bollinger Bands
    bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
    return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iBands      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &base_values[],     // le tampon d'indicateur de la
                           double &upper_values[],    // le tampon d'indicateur de la
                           double &lower_values[],    // le tampon d'indicateur de la
                           int shift,                 // le décalage
                           int ind_handle,            // le handle de l'indicateur iBa
                           int amount                 // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
    ResetLastError();
//--- remplissons la partie du tableau MiddleBuffer par les valeurs du tampon d'indica
    if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
      {
       //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
       PrintFormat("n'a pas réussi à copier les données de l'indicateur iBands, le code
       //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
       return(false);
      }
```

```
//--- remplissons la partie du tableau UpperBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("n'a pas réussi à copier les données de l'indicateur iBands, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau LowerBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("n'a pas réussi à copier les données de l'indicateur iBands, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iBullsPower

Rend le handle de l'indicateur Bulls Power. Seulement un tampon.

```
int  iBullsPower(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 ma_period,         // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBullsPower.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iBullsPowe
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iBullsPower
#property indicator_label1  "iBullsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBullsPower,       // utiliser iBullsPower
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iBullsPower;  // le type de la fonction
input int                  ma_period=13;           // la période de la glissante
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- le tampon d'indicateur
double         iBullsPowerBuffer[];
//--- la variable pour stocker le handle de l'indicateur iBullsPower
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Bulls Power
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iBullsPowerBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iBullsPower)
      handle=iBullsPower(name,period,ma_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- la période de la glissante
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro d'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iBullsPower pou
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Bulls Power a été
   short_name=StringFormat("iBullsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iBullsPower
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

```
      {
       //--- si le tableau iBullsPowerBuffer est plus grand, que les valeurs dans l'inc
       //--- autrement copions moins que la taille des tampons d'indicateur
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- signifie que notre indicateur est calculé non pour la première fois et dès
       //--- pour le calcul a été ajouté pas plus d'une barre
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- remplissons le tableau iBullsPowerBuffer  par les valeurs de l'indicateur Bulls
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
    if(!FillArrayFromBuffer(iBullsPowerBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
    string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- déduisons le message de service sur le graphique
    Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Bulls Power
    bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
    return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iBullsPower   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Bul
                         int ind_handle,    // le handle de l'indicateur iBullsPower
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
    ResetLastError();
//--- remplissons la partie du tableau iBullsPowerBuffer par les valeurs du tampon d'i
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
      {
       //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
       PrintFormat("On n'a pas réussi à copier les données de l'indicateur iBullsPower,
       //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
       return(false);
      }
//--- tout a réussi
    return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

iCCI

Rend le handle de l'indicateur Commodity Channel Index. Seulement un tampon.

```
int  iCCI(
   string               symbol,            // nom du symbole
   ENUM_TIMEFRAMES      period,            // période
   int                  ma_period,         // période de la prise de moyenne
   ENUM_APPLIED_PRICE   applied_price      // type du prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iCCI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iCCI."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
```

```
#property indicator_plots   1
//--- la construction iCCI
#property indicator_label1  "iCCI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iCCI,              // utiliser iCCI
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iCCI;               // le type de la fonction
input int                  ma_period=14;                 // la période de la glissante
input ENUM_APPLIED_PRICE   applied_price=PRICE_TYPICAL;  // le type du prix
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- le tampon d'indicateur
double         iCCIBuffer[];
//--- la variable pour stocker le handle de l'indicateur iCCI
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Commodity Channel Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
```

```
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iCCI)
      handle=iCCI(name,period,ma_period,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le type du prix
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_CCI,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iCCI pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Bollinger Bands® a
   short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- le nombre de valeurs copiées de l'indicateur iCCI
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iATRBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iCCIBuffer   par les valeurs de l'indicateur Commodity Ch
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Commodity Channel Index
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iCCI          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Comm
                         int ind_handle,    // le handle de l'indicateur iCCI
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
```

© 2000-2024, MetaQuotes Ltd.

```
   ResetLastError();
//--- remplissons la partie du tableau iCCIBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iCCI, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iChaikin

Rend le handle de l'indicateur Chaikin Oscillator. Seulement un tampon.

```
int  iChaikin(
   string                symbol,              // nom du symbole
   ENUM_TIMEFRAMES       period,              // période
   int                   fast_ma_period,      // période rapide
   int                   slow_ma_period,      // période lente
   ENUM_MA_METHOD        ma_method,           // type de lissage
   ENUM_APPLIED_VOLUME   applied_volume       // volume utilisé
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

fast_ma_period

[in] La période rapide de la prise de moyenne pour le calcul de l'indicateur.

slow_ma_period

[in] La période lente de la prise de moyenne pour le calcul de l'indicateur.

ma_method

[in] Le type de la prise de moyenne. Peut être chacune des constantes de la prise de moyenne ENUM_MA_METHOD.

applied_volume

[in] Le volume utilisé. Peut être chacun de l'énumération ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iChaikin.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
```

```
#property description "des tampons d'indicateur pour l'indicateur technique iChaikin."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iChaikin
#property indicator_label1  "iChaikin"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iChaikin,          // utiliser iChaikin
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iChaikin;           // le type de la fonction
input int                  fast_ma_period=3;             // la période rapide
input int                  slow_ma_period=10;            // la période lente
input ENUM_MA_METHOD       ma_method=MODE_EMA;           // le type du lissage
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // le type du volume
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- le tampon d'indicateur
double         iChaikinBuffer[];
//--- la variable pour stocker le handle de l'indicateur iChaikin
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Chaikin Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iChaikin)
      handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_volu
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période rapide
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ma_period;
      //--- la période lente
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ma_period;
      //--- le type du lissage
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- le type du volume
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iChaikin pour l
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Chaikin Oscillator
   short_name=StringFormat("iChaikin(%s/%s, %d, %d, %s, %s)",name,EnumToString(period)
                           fast_ma_period,slow_ma_period,
                           EnumToString(ma_method),EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iChaikin
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iChaikinBuffer est plus grand, que les valeurs dans l'indica
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iChaikinBuffer  par les valeurs de l'indicateur Chaikin O
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Chaikin Oscillator
```

© 2000-2024, MetaQuotes Ltd.

```
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iChaikin      |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Chai
                         int ind_handle,    // le handle de l'indicateur iChaikin
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iChaikinBuffer par les valeurs du tampon d'indi
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iChaikin, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera con
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iCustom

Rend le handle de l'indicateur indiqué d'utilisateur.

```
int  iCustom(
   string           symbol,      // nom du symbole
   ENUM_TIMEFRAMES  period,      // période
   string           name         // dossier/le nom de_l'indicateur d'utilisateur
   ...                           // liste des paramètres d'entrée de l'indicateur
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

name

[in] Le nom de l'indicateur d'utilisateur contenant le chemin relativement le directoire radical des indicateurs (MQL5/Indicators/). Si l'indicateur se trouve au sous-répertoire, par exemple, dans MQL5/Indicators/**Examples**, son nom doit être spécifié comme - "**Examples**\\le nom_de l'indicateur" (l'indication de l'inverse double slash au lieu de slash simple comme le délimiteur).

...

[in] les paramètres input de l'indicateur d'utilisateur divisés par les virgules. Le type et l'héritage des paramètres doit correspondre. Si les paramètres ne sont pas indiqués, on utilise les valeurs par défaut.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE.

Note

L'indicateur d'utilisateur doit être compilé (le fichier avec l'extension EX5) et se trouver dans le directoire *MQL5/Indicators* du terminal de client ou le sous-répertoire inclus.

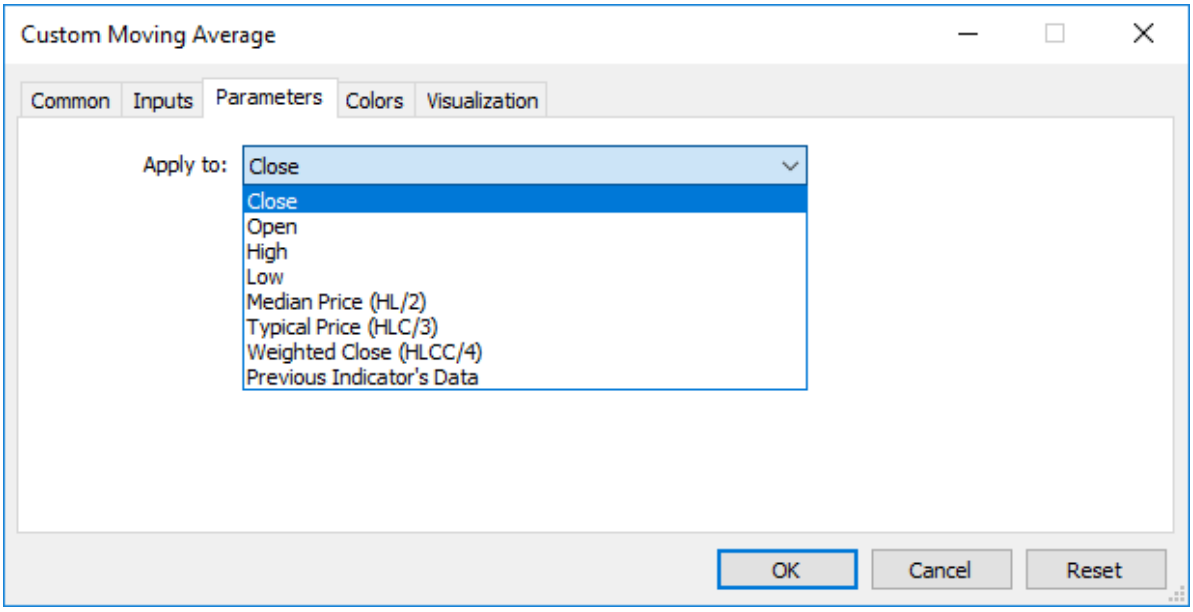
Les indicateurs nécessaires au test sont définis automatiquement de l'appel des fonctions, iCustom(), si le paramètre correspondant est spécifié par la ligne constante. Pour les autres cas (l'utilisation de la fonction IndicatorCreate() ou l'utilisation de la ligne non constante dans le paramètre qui spécifie le nom de l'indicateur) la propriété donnée #property tester_indicator est nécessaire:

```
#property tester_indicator "indicator_name.ex5"
```

Si dans l'indicateur on utilise la première forme de l'appel, au lancement de l'indicateur d'utilisateur sur l'onglet "Parameters" on peut indiquer supplémentairement sur quelles données il sera calculé. Si

© 2000-2024, MetaQuotes Ltd.

le paramètre "Apply to" n'est pas choisi évidemment, par défaut le calcul est produit selon les valeurs "Close".



A l'appel de l'indicateur d'utilisateur du programme mql5 le paramètre Applied_Price ou le handle d'un autre indicateur doit être transmis le dernier après toutes les variables d'entrée prévues par l'indicateur d'utilisateur.

Voir aussi

Propriétés des programmes, Accès aux séries temporelles et les données des indicateurs, IndicatorCreate(), IndicatorRelease()

Exemple:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int MA_Period=21;
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- indicator buffers
double         Label1Buffer[];
//--- handle de l'indicateur d'utilisateur Custom Moving Average.mq5
int MA_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
   ResetLastError();
   MA_handle=iCustom(NULL,0,"Examples\\Custom Moving Average",
                     MA_Period,
                     MA_Shift,
                     MA_Method,
                     PRICE_CLOSE // calculons selon les prix de la clôture
                     );
   Print("MA_handle = ",MA_handle,"  error = ",GetLastError());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- copions les valeurs de l'indicateur Custom Moving Average à notre tampon d'indic
   int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
   Print("copy = ",copy,"    rates_total = ",rates_total);
//--- si la tentative est ratée - signalez-le
   if(copy<=0)
      Print("La tentative ratée de recevoir les valeurs de l'indicateur  Custom Moving
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

iDEMA

Rend le handle de l'indicateur Double Exponential Moving Average. Seulement un tampon.

```
int  iDEMA(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 ma_period,         // période de la prise de moyenne
   int                 ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_APPLIED_PRICE  applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iDEMA.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iDEMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
```

```
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iDEMA
#property indicator_label1  "iDEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDEMA,             // utiliser iDEMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iDEMA;           // le type de la fonction
input int                  ma_period=14;              // la période de la glissante mo
input int                  ma_shift=0;                // le décalage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iDEMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iDEMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Double Exponential Moving Aver
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
   //---spécifions le décalage
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
```

© 2000-2024, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iDEMA)
      handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[3];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du prix
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iDEMA pour la p
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Double Exponential
   short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iDEMA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iDEMABuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iDEMABuffer  par les valeurs de l'indicateur Double Expor
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//---retiendrons le nombre de valeurs dans l'indicateur Double Exponential Moving Aver
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iDEMA         |
```

```
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Doub
                         int shift,         // le décalage
                         int ind_handle,    // le handle de l'indicateur iDEMA
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iDEMABuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iDEMA, le co
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera con
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iDeMarker

Rend le handle de l'indicateur DeMarker. Seulement un tampon.

```
int  iDeMarker(
   string           symbol,        // nom du symbole
   ENUM_TIMEFRAMES  period,        // période
   int              ma_period      // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iDeMarker.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iDeMarker.
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iDeMarker
#property indicator_label1  "iDeMarker"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  0.3
#property indicator_level2  0.7
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDeMarker,          // utiliser iDeMarker
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iDeMarker;       // le type de la fonction
input int                  ma_period=14;              // la période de la glissante
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iDeMarkerBuffer[];
//--- la variable pour stocker le handle de l'indicateur iDeMarker
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur DeMarker
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iDeMarker)
      handle=iDeMarker(name,period,ma_period);
```

© 2000-2024, MetaQuotes Ltd.

```
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iDeMarker pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur DeMa
   short_name=StringFormat("iDeMarker(%s/%s, period=%d)",name,EnumToString(period),ma_
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iDeMarker
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
```

```
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iDEMABuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iDeMarkerBuffer par les valeurs de l'indicateur DeMarker
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur DeMarker
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iDeMarker     |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs DeMa
                         int ind_handle,    // le handle de l'indicateur iDeMarker
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iDeMarkerBuffer par les valeurs du tampon d'inc
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iDeMarker, l
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iEnvelopes

Rend le handle de l'indicateur Envelopes.

```
int  iEnvelopes(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 ma_period,         // période pour le calcul de la ligne moyenn
   int                 ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_MA_METHOD      ma_method,         // type de lissage
   ENUM_APPLIED_PRICE  applied_price,     // type de prix ou le handle
   double              deviation          // divergence des frontières de la ligne moy
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne de la ligne principale de l'indicateur.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique des prix.

ma_method

[in] La méthode de la prise de moyenne. Peut être chacune des valeurs de l'énumération ENUM_MA_METHOD.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

deviation

[in] La divergence de la ligne principale en pourcentage.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - UPPER_LINE, 1 - LOWER_LINE.

Exemple:

```
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//|                                                  Demo_iEnvelopes.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iEnvelopes
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- la construction Upper
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Lower
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iEnvelopes,        // utiliser iEnvelopes
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iEnvelopes;      // le type de la fonction
input int                  ma_period=14;              // la période de la glissante
input int                  ma_shift=0;                // le décalage
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // le type du lissage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input double               deviation=0.1;             // le divergence des frontières
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         UpperBuffer[];
double         LowerBuffer[];
//--- la variable pour stocker le handle de l'indicateur iEnvelopes
```

```
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Envelopes
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- spécifions le décalage pour chaque ligne
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iEnvelopes)
      handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviati
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[5];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du lissage
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      //--- le type du prix
```

© 2000-2024, MetaQuotes Ltd.

```
      pars[4].type=TYPE_DOUBLE;
      pars[4].double_value=deviation;
      handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iEnvelopes pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Enve
   short_name=StringFormat("iEnvelopes(%s/%s, %d, %d, %s,%s, %G)",name,EnumToString(pe
   ma_period,ma_shift,EnumToString(ma_method),EnumToString(applied_price),deviation);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iEnvelopes
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

```
      //--- si le tableau UpperBuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux UpperBuffer et LowerBuffer par les valeurs de l'indicat
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy))
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Envelopes
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iEnvelopes  |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &upper_values[],    // le tampon d'indicateur de la
                           double &lower_values[],    // le tampon d'indicateur de la
                           int shift,                 // le décalage
                           int ind_handle,            // le handle de l'indicateur iEn
                           int amount                 // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau UpperBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,-shift,amount,upper_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iEnvelopes,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- remplissons la partie du tableau LowerBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iEnvelopes,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iForce

Rend le handle de l'indicateur Force Index. Seulement un tampon.

```
int  iForce(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 ma_period,         // période de la prise de moyenne
   ENUM_MA_METHOD      ma_method,         // type de lissage
   ENUM_APPLIED_VOLUME applied_volume     // type du volume pour le calcul
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

ma_method

[in] La méthode de la prise de moyenne. Peut être chacune des valeurs de l'énumération ENUM_MA_METHOD.

applied_volume

[in] Le volume utilisé. Peut être chacun des valeurs de l'énumération ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iForce.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iForce."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
```

```
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iForce
#property indicator_label1  "iForce"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iForce,            // utiliser iForce
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iForce;             // le type de la fonction
input int                  ma_period=13;                 // la période de la prise de
input ENUM_MA_METHOD       ma_method=MODE_SMA;           // le type du lissage
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // le type du volume
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- le tampon d'indicateur
double         iForceBuffer[];
//--- la variable pour stocker le handle de l'indicateur iForce
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Force
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
```

© 2000-2024, MetaQuotes Ltd.

```
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iForce)
      handle=iForce(name,period,ma_period,ma_method,applied_volume);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[3];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le type du lissage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_method;
      //--- le type du volume
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_volume;
      //--- le type du prix
      handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iForce pour la
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Forc
   short_name=StringFormat("iForce(%s/%s, %d, %s, %s)",name,EnumToString(period),
                           ma_period,EnumToString(ma_method),EnumToString(applied_volu
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iForce
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iForceBuffer est plus grand, que les valeurs dans l'indicate
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iForceBuffer par les valeurs de l'indicateur Force
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Force
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iForce        |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Forc
                         int ind_handle,    // le handle de l'indicateur iForce
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iForceBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iForce, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iFractals

Rend le handle de l'indicateur Fractals.

```
int  iFractals(
   string           symbol,      // nom du symbole
   ENUM_TIMEFRAMES  period       // période
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - UPPER_LINE, 1 - LOWER_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iFractals.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iFractals.
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- la construction FractalUp
```

```
#property indicator_label1  "FractalUp"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
//--- la construction FractalDown
#property indicator_label2  "FractalDown"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFractals,          // utiliser iFractals
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation              type=Call_iFractals;          // le type de la fonction
input string                symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // le temps trame
//--- les tampons d'indicateur
double          FractalUpBuffer[];
double          FractalDownBuffer[];
//--- la variable pour stocker le handle de l'indicateur iFractals
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Fractals
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
//--- spécifions les codes par le symbole de l'ensemble Wingdings pour les propriétés
   PlotIndexSetInteger(0,PLOT_ARROW,217); // la flèche en haut
   PlotIndexSetInteger(1,PLOT_ARROW,218); // la flèche en bas
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
```

```
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iFractals)
      handle=iFractals(name,period);
   else
      handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iFractals pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Frac
   short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iFractals
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
```

```
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau FractalUpBuffer est plus grand, que les valeurs dans l'indic
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux FractalUpBuffer et FractalDownBuffer par les valeurs de
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy))
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Fractals
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iFractals   |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &up_arrows[],        // le tampon d'indicateur des f
                           double &down_arrows[],      // le tampon d'indicateur des f
                           int ind_handle,             // le handle de l'indicateur iF
                           int amount                  // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//---remplissons la partie du tableau FractalUpBuffer par les valeurs du tampon d'indi
   if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iFractals au
                  GetLastError());
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
```

```
//--- remplissons la partie du tableau FractalDownBuffer par les valeurs du tampon d'i
   if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iFractals da
                  GetLastError());
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera con
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iFrAMA

Rend le handle de l'indicateur Fractal Adaptive Moving Average. Seulement un tampon.

```
int  iFrAMA(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 ma_period,         // période de la prise de moyenne
   int                 ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_APPLIED_PRICE  applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique des prix.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iFrAMA.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iFrAMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
```

```
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iFrAMA
#property indicator_label1  "iFrAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFrAMA,            // utiliser iFrAMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iFrAMA;             // le type de la fonction
input int                  ma_period=14;                 // la période de la prise de
input int                  ma_shift=0;                   // le décalage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;    // le type du prix
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- le tampon d'indicateur
double         iFrAMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iFrAMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Fractal Adaptive Moving Averag
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
//--- spécifions le décalage
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
```

© 2000-2024, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iFrAMA)
      handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[3];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du prix
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      //--- le type du prix
      handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iFrAMA pour la
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur iFr
   short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2024, MetaQuotes Ltd.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iFrAMA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iFrAMABuffer est plus grand, que les valeurs dans l'indicate
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iFrAMABuffer  par les valeurs de l'indicateur Fractal Ada
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur Fractal Adaptive Moving Averag
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Remplissons le tampon d'indicateur de l'indicateur iFrAMA        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Frac
                         int shift,         // le décalage
                         int ind_handle,    //le handle de l'indicateur iFrAMA
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iFrAMABuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iFrAMA, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iGatore

Rend le handle de l'indicateur Gator. L'Oscillateur montre la différence entre les lignes bleues et rouges de l'Alligator (l'histogramme supérieur) et la différence entre la ligne rouge et verte (l'histogramme inférieur).

```
int  iGator(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 jaw_period,        // période pour le calcul des mâchoires
   int                 jaw_shift,         // décalage horizontal des mâchoires
   int                 teeth_period,      // période pour le calcul des dents
   int                 teeth_shift,       // décalage horizontal des dents
   int                 lips_period,       // période pour le calcul des lèvres
   int                 lips_shift,        // décalage horizontal des lèvres
   ENUM_MA_METHOD      ma_method,         // type de lissage
   ENUM_APPLIED_PRICE  applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

jaw_period

[in] La période de la prise de moyenne de la ligne bleue (la mâchoire d'alligator).

jaw_shift

[in] Le décalage de la ligne bleue de l'Alligator par rapport au graphique du prix. Il n'est pas directement raccordé avec le changement visuel de l'histogramme d'indicateur.

teeth_period

[in] La période de la prise de moyenne de la ligne rouge (les dents d'alligator).

teeth_shift

[in] Le décalage de la ligne rouge d'Alligator par rapport au graphique du prix. Il n'est pas directement raccordé avec le changement visuel de l'histogramme d'indicateur.

lips_period

[in] La période de la prise de moyenne de la ligne verte (les lèvres d'alligator).

lips_shift

[in] Le décalage de la ligne verte de l'Alligator par rapport au graphique du prix. Il n'est pas directement raccordé avec le changement visuel de l'histogramme d'indicateur.

ma_method

[in] La méthode de la prise de moyenne. Peut être l'une des valeurs ENUM_MA_METHOD.

© 2000-2024, MetaQuotes Ltd.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - UPPER_HISTOGRAM, 1- le tampon coloré de l'histogramme supérieur, 2 - LOWER_HISTOGRAM, 3- le tampon coloré de l'histogramme inférieur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iGator.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iGator."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Gator Oscillator standard

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
//--- la construction GatorUp
#property indicator_label1  "GatorUp"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction GatorDown
#property indicator_label2  "GatorDown"
#property indicator_type2   DRAW_COLOR_HISTOGRAM
#property indicator_color2  clrGreen, clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
```

```
  {
   Call_iGator,            // utiliser iGator
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iGator;       // le type de la fonction
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
input int                  jaw_period=13;          // la période pour la ligne des Mâc
input int                  jaw_shift=8;            // le décalage de la ligne des Mâch
input int                  teeth_period=8;         // la période pour la ligne des Der
input int                  teeth_shift=5;          // le décalage de la ligne des Mâch
input int                  lips_period=5;          // la période pour la ligne des Lèv
input int                  lips_shift=3;           // le décalage de la ligne des Lèvr
input ENUM_MA_METHOD       MA_method=MODE_SMMA;    // la méthode de la prise de moyenn
input ENUM_APPLIED_PRICE   applied_price=PRICE_MEDIAN;// le type de prix utilisé pour
//--- les tampons d'indicateur
double         GatorUpBuffer[];
double         GatorUpColors[];
double         GatorDownBuffer[];
double         GatorDownColors[];
//--- la variable pour stocker le handle de l'indicateur iGator
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- les valeurs des décalages pour l'histogramme supérieur et inférieur
int shift;
//--- stockerons le nombre de valeurs dans l'indicateur Gator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
   SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);
/*
  Tous les décalages indiqués dans les paramètres se rapportent à l'indicateur Alligat
  C'est pourquoi les décalages indiqués ne produisent pas le décalage de l'indicateur
  dont les valeurs sont utilisées pour le calcul de Gator Oscillator!
*/
//--- Nous calculerons le décalage pour les histogrammes supérieurs et inférieurs, qui
   shift=MathMin(jaw_shift,teeth_shift);
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- malgré le fait qu'il y a deux histogrammes dans l'indicateur, on utilise le déca
   PlotIndexSetInteger(1,PLOT_SHIFT,shift);

//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iGator)
      handle=iGator(name,period,jaw_period,jaw_shift,teeth_period,teeth_shift,
                    lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[8];
      //--- les périodes et les décalages des lignes de l'Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
      //--- le type du lissage
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
      //--- le type du prix
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
      //--- créons le handle
      handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iGator pour la
```

© 2000-2024, MetaQuotes Ltd.

```
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Gator Oscillator a
   short_name=StringFormat("iGator(%s/%s, %d, %d ,%d, %d, %d, %d)",name,EnumToString(p
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iGator
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//---si c'est un premier lancement des calculs de notre indicateur ou a changé le nomb
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau GatorUpBuffer est plus grand, que les valeurs dans l'indicat
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
```

```
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur Gator Oscillator
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownCol
      shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Gator Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iGator      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &ups_buffer[],         // le tampon d'indicateur pou
                           double &up_color_buffer[],    // le tampon d'indicateur pou
                           double &downs_buffer[],       // le tampon d'indicateur pou
                           double &downs_color_buffer[], // le tampon d'indicateur pou
                           int u_shift,                  // le décalage pour l'histogr
                           int ind_handle,               // le handle de l'indicateur
                           int amount                    // le nombre de valeurs copié
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- emplissons la partie du tableau GatorUpBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iGator, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- emplissons la partie du tableau GatorUpColors par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iGator, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
```

```
//--- remplissons la partie du tableau GatorDownBuffer par les valeurs du tampon d'inc
   if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iGator, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau GatorDownColors par les valeurs du tampon d'inc
   if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iGator, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iIchimoku

Rend le handle de l'indicateur Ichimoku Kinko Hyo.

```
int  iIchimoku(
   string           symbol,            // nom du symbole
   ENUM_TIMEFRAMES  period,            // période
   int              tenkan_sen,        // période Tenkan-sen
   int              kijun_sen,         // période Kijun-sen
   int              senkou_span_b      // période Senkou Span B
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

tenkan_sen

[in] La période de la prise de moyenne Tenkan Sen.

kijun_sen

[in] La période de la prise de moyenne Kijun Sen.

senkou_span_b

[in] La période de la prise de moyenne Senkou Span B.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - TENKANSEN_LINE, 1 - KIJUNSEN_LINE, 2 - SENKOUSPANA_LINE, 3 - SENKOUSPANB_LINE, 4 - CHIKOUSPAN_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iIchimoku.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
```

```
#property description "des tampons d'indicateur pour l'indicateur technique iIchimoku.
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Ichimoku Kinko Hyo standa

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   4
//--- la construction Tenkan_sen
#property indicator_label1  "Tenkan_sen"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Kijun_sen
#property indicator_label2  "Kijun_sen"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrBlue
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- la construction Senkou_Span
#property indicator_label3  "Senkou Span A;Senkou Span B" // dans Data Window deux cha
#property indicator_type3   DRAW_FILLING
#property indicator_color3  clrSandyBrown, clrThistle
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//--- la construction Chikou_Span
#property indicator_label4  "Chinkou_Span"
#property indicator_type4   DRAW_LINE
#property indicator_color4  clrLime
#property indicator_style4  STYLE_SOLID
#property indicator_width4  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iIchimoku,          // utiliser iIchimoku
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation              type=Call_iIchimoku;       // le type de la fonction
input int                   tenkan_sen=9;              // la période Tenkan-sen
input int                   kijun_sen=26;              // la période Kijun-sen
input int                   senkou_span_b=52;          // la période Senkou Span B
input string                symbol=" ";                // le symbole
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
double         Tenkan_sen_Buffer[];
double         Kijun_sen_Buffer[];
double         Senkou_Span_A_Buffer[];
double         Senkou_Span_B_Buffer[];
double         Chinkou_Span_Buffer[];
//--- la variable pour stocker le handle de l'indicateur iIchimoku
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Ichimoku Kinko Hyo
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,Tenkan_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(1,Kijun_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(2,Senkou_Span_A_Buffer,INDICATOR_DATA);
   SetIndexBuffer(3,Senkou_Span_B_Buffer,INDICATOR_DATA);
   SetIndexBuffer(4,Chinkou_Span_Buffer,INDICATOR_DATA);
//--- spécifions les décalages pour le canal Senkou Span sur kijun_sen bars au futur
   PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//---il ne faut pas spécifier le décalage pour la ligne Chikou Span, parce que les don
//--- sont stockées dans l'indicateur iIchimoku déjà avec le décalage
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iIchimoku)
      handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[3];
      //--- les périodes et les décalages des lignes de l'Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=tenkan_sen;
```

© 2000-2024, MetaQuotes Ltd.

```
      pars[1].type=TYPE_INT;
      pars[1].integer_value=kijun_sen;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=senkou_span_b;
      //--- créons le handle
      handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iIchimoku pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Ichi
   short_name=StringFormat("iIchimoku(%s/%s, %d, %d ,%d)",name,EnumToString(period),
                           tenkan_sen,kijun_sen,senkou_span_b);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iIchimoku
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
```

```
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau Tenkan_sen_Buffer est plus grand, que les valeurs dans l'inc
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur Ichimoku Kinko Hyo
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer,S
      kijun_sen,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Ichimoku Kinko Hyo
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iIchimoku   |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[],     // le tampon d'indicateur
                           double &kijun_sen_buffer[],      // le tampon d'indicateur
                           double &senkou_span_A_buffer[],  // le tampon d'indicateur
                           double &senkou_span_B_buffer[],  // le tampon d'indicateur
                           double &chinkou_span_buffer[],   // le tampon d'indicateur
                           int senkou_span_shift,           // le décalage des lignes
                           int ind_handle,                  // le handle de l'indicate
                           int amount                       // le nombre de valeurs co
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau Tenkan_sen_Buffer par les valeurs du tampon d'i
   if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
```

```
      PrintFormat("1.On n'a pas réussi à copier les données de l'indicateur iGator, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau Kijun_sen_Buffer par les valeurs du tampon d'in
   if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("2..On n'a pas réussi à copier les données de l'indicateur iGator, l
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau Chinkou_Span_Buffer par les valeurs du tampon d
//--- à senkou_span_shift>0 la ligne est décalée au futur sur senkou_span_shift bars
   if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("3.On n'a pas réussi à copier les données de l'indicateur iGator, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau Senkou_Span_A_Buffer par les valeurs du tampon
//--- à senkou_span_shift>0 la ligne est décalée au futur sur senkou_span_shift bars
   if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("4.On n'a pas réussi à copier les données de l'indicateur iGator, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau Senkou_Span_B_Buffer par les valeurs du tampon
//--- il ne faut pas prendre en considération le décalage pendant le copiage Chinkou S
//--- sont stockées dans l'indicateur iIchimoku déjà avec le décalage
   if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("5.On n'a pas réussi à copier les données de l'indicateur iGator, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iBWMFI

Rend le handle de l'indicateur Market Facilitation Index. Seulement un tampon.

```
int  iBWMFI(
   string                symbol,              // nom du symbole
   ENUM_TIMEFRAMES       period,              // période
   ENUM_APPLIED_VOLUME   applied_volume       // type de volume pour le calcul
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

applied_volume

[in] Le volume utilisé. Peut être chacun des valeurs de l'énumération ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iBWMFI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iBWMFI."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- la construction iBWMFI
#property indicator_label1  "iBWMFI"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrLime,clrSaddleBrown,clrBlue,clrPink
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBWMFI,            // utiliser iBWMFI
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iBWMFI;          // le type de la fonction
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;// / le type du volume
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iBWMFIBuffer[];
double         iBWMFIColors[];
//--- la variable pour stocker le handle de l'indicateur iBWMFI
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Market Facilitation Index de B
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iBWMFI)
      handle=iBWMFI(name,period,applied_volume);
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- remplirons la structure par les valeurs des paramètres de l'indicateur
       MqlParam pars[1];
       //--- le type du volume
       pars[0].type=TYPE_INT;
       pars[0].integer_value=applied_volume;
       handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
      }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
      {
       //--- informons sur l'échec et déduisons le numéro de l'erreur
       PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iIchimoku pour
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- le travail de l'indicateur est terminé avant terme
       return(INIT_FAILED);
      }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Mar
   short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iBWMFI
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
      {
       PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
       return(0);
      }
```

```
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau Tenkan_sen_Buffer est plus grand, que les valeurs dans l'inc
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- emplissons les tableaux par les valeurs de l'indicateur Market Facilitation Inde
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy)) return
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Market Facilitation Index de
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iBWMFI      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &values[],    // le tampon d'indicateur des valeurs
                           double &colors[],    // le tampon d'indicateur des couleurs
                           int ind_handle,      // le handle de l'indicateur iBWMFI
                           int amount           // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iBWMFIBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iBWMFI, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
```

```
//--- remplissons la partie du tableau iBWMFIColors par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iBWMFI, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iMomentum

Rend le handle de l'indicateur Momentum. Seulement un tampon.

```
int  iMomentum(
   string                symbol,            // nom du symbole
   ENUM_TIMEFRAMES       period,            // période
   int                   mom_period,        // période de la prise de moyenne
   ENUM_APPLIED_PRICE    applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

mom_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iMomentum.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iMomentum.
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Momentum standard."

#property indicator_separate_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iMomentum
#property indicator_label1  "iMomentum"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMomentum,         // utiliser iMomentum
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iMomentum;       // le type de la fonction
input int                  mom_period=14;             // la période du Momentum
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iMomentumBuffer[];
//--- la variable pour stocker le handle de l'indicateur iMomentum
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Momentum
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- créons le handle de l'indicateur
   if(type==Call_iMomentum)
      handle=iMomentum(name,period,mom_period,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=mom_period;
      //--- le type du prix
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iMomentum pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Mome
   short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                           mom_period, EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iMomentum
```

© 2000-2024, MetaQuotes Ltd.

```
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iMomentumBuffer est plus grand, que les valeurs dans l'indic
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iMomentumBuffer par les valeurs de l'indicateur Momentum
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- etiendrons le nombre de valeurs dans l'indicateur Momentum
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iMomentum     |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Mome
                         int ind_handle,    // le handle de l'indicateur iMomentum
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iMomentumBuffer par les valeurs du tampon d'inc
```

```
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iMomentum, 
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera co
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iMFI

La calcul Money Flow Index.

```
int  iMFI(
   string                 symbol,              // nom du symbole
   ENUM_TIMEFRAMES        period,              // période
   int                    ma_period,           // période de la prise de moyenne
   ENUM_APPLIED_VOLUME    applied_volume       // type de volume pour le calcul
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

applied_volume

[in] Le volume utilisé. Peut être l'un des ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iMFI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iMFI."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Money Flow Index standard

#property indicator_separate_window
#property indicator_buffers 1
```

```
#property indicator_plots   1
//--- la construction iMFI
#property indicator_label1  "iMFI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  20
#property indicator_level2  80
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMFI,              // utiliser iMFI
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iMFI;               // le type de la fonction
input int                  ma_period=14;                 // la période
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // le type du volume
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- le tampon d'indicateur
double         iMFIBuffer[];
//--- la variable pour stocker le handle de l'indicateur iMFI
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Money Flow Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
```

```
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iMFI)
      handle=iMFI(name,period,ma_period,applied_volume);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le type du volume
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_MFI,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iMFI pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Mone
   short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period, EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- le nombre de valeurs copiées de l'indicateur iMFI
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iMFIBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iMFIBuffer par les valeurs de l'indicateur Money Flow Inc
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Money Flow Index
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iMFI          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Mone
                         int ind_handle,    // le handle de l'indicateur iMFI
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
```

```
   ResetLastError();
//--- remplissons la partie du tableau iMFIBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iMFI, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iMA

Rend le handle de l'indicateur du moyen glissant. Seulement un tampon.

```
int  iMA(
   string                symbol,            // nom du symbole
   ENUM_TIMEFRAMES       period,            // période
   int                   ma_period,         // période de la prise de moyenne
   int                   ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_MA_METHOD        ma_method,         // type de lissage
   ENUM_APPLIED_PRICE    applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul du moyen glissant.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

ma_method

[in] La méthode de la prise de moyenne. Peut être l'une des valeurs ENUM_MA_METHOD.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
```

```
#property description "des tampons d'indicateur pour l'indicateur technique iMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Moving Average standard."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iMA
#property indicator_label1  "iMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMA,                // utiliser iMA
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iMA;                // le type de la fonction
input int                  ma_period=10;                 // la période de la moyenne
input int                  ma_shift=0;                   // le décalage
input ENUM_MA_METHOD       ma_method=MODE_SMA;           // le type du lissage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;    // le type du prix
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- le tampon d'indicateur
double         iMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//---stockerons le nombre de valeurs dans l'indicateur Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);
//--- spécifions le décalage
```

```
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iMA)
      handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du lissage
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MA,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iMA pour la pai
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Moving Average a é
   short_name=StringFormat("iMA(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period, ma_shift,EnumToString(ma_method),EnumToString(ap
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iMA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- - si c'est un premier lancement des calculs de notre indicateur ou a changé le r
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iMABuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iMABuffer par les valeurs de l'indicateur Moving Average
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
```

© 2000-2024, MetaQuotes Ltd.

```
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Moving Average
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iMA           |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // le tampon d'indicateur des valeurs Mov
                         int shift,          // le décalage
                         int ind_handle,     // le handle de l'indicateur iMA
                         int amount          // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iMABuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iMA, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iOsMA

Rend le handle de l'indicateur Moving Average of Oscillator. L'oscillateur OsMA montre la différence entre les valeurs MACD et sa ligne d'alarme. Seulement un tampon.

```
int  iOsMA(
   string              symbol,              // nom du symbole
   ENUM_TIMEFRAMES     period,              // période
   int                 fast_ema_period,     // période de la moyenne rapide
   int                 slow_ema_period,     // période de la moyenne lente
   int                 signal_period,       // période de la prise de moyenne de la di
   ENUM_APPLIED_PRICE  applied_price        // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

fast_ema_period

[in] La période de la prise de moyenne pour le calcul de la moyenne rapide glissant.

slow_ema_period

[in] La période de la prise de moyenne pour le calcul de la moyenne lente glissant.

signal_period

[in] La période de la prise de moyenne pour le calcul de la ligne d'alarme.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Dans certains systèmes cet oscillateur s'appelle l'histogramme MACD.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOsMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iOsMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description ""Tous les autres paramètres comme dans Moving Average of Oscill

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iOsMA
#property indicator_label1  "iOsMA"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOsMA,             // utiliser iOsMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iOsMA;           // le type de la fonction
input int                  fast_ema_period=12;        // la période de la moyenne rapi
input int                  slow_ema_period=26;        // la période de la moyenne lent
input int                  signal_period=9;           // la période de la prise de moy
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iOsMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iAMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Moving Average of Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
```

```
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iOsMA)
      handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période rapide
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- la période lente
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- la période de la prise de moyenne de la différence entre les moyennes rapi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iOsMA pour la p
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Moving Average of
   short_name=StringFormat("iOsMA(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
```

```
                              fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iOsMA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iOsMABuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iOsMA
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

```
                              short_name,
                              values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur  Moving Average of Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iOsMA                 |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // le tampon d'indicateur des valeurs
                         int ind_handle,        // le handle de l'indicateur iOsMA
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iOsMABuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iOsMA, le co
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iMACD

Rend le handle de l'indicateur Moving Averages Convergence/Divergence. Dans ces systèmes, où on appelle OsMA l'histogramme MACD, cet indicateur est représenté en forme de deux lignes. Dans le terminal de client la convergence/divergence des moyens glissant se dessine en forme de l'histogramme.

```
int  iMACD(
   string              symbol,              // nom du symbole
   ENUM_TIMEFRAMES     period,              // période
   int                 fast_ema_period,     // période de la moyenne rapide
   int                 slow_ema_period,     // période de la moyenne lente
   int                 signal_period,       // période de la prise de moyenne de la di
   ENUM_APPLIED_PRICE  applied_price        // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

fast_ema_period

[in] La période de la prise de moyenne pour le calcul de la moyenne rapide glissant.

slow_ema_period

[in] La période de la prise de moyenne pour le calcul de la moyenne lente glissant.

signal_period

[in] La période de la prise de moyenne pour le calcul de la ligne d'alarme.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des bufers : 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iMACD.mq5 |
```

© 2000-2024, MetaQuotes Ltd.

```
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iMACD."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans MACD standard."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- la construction MACD
#property indicator_label1  "MACD"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_DOT
#property indicator_width2  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMACD,             // utiliser iMACD
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iMACD;           // le type de la fonction
input int                  fast_ema_period=12;        // la période de la moyenne rapi
input int                  slow_ema_period=26;        // la période de la moyenne lent
input int                  signal_period=9;           // la période de la prise de moy
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- les tampons d'indicateur
double         MACDBuffer[];
double         SignalBuffer[];
//--- la variable pour stocker le handle de l'indicateur iMACD
int    handle;
```

```
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Moving Averages Convergence/Di
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iMACD)
      handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période rapide
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- la période lente
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- la période de la prise de moyenne de la différence entre les moyennes rapi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MACD,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iMACD pour la p
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Moving Averages Co
   short_name=StringFormat("iMACD(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
                           fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iMACD
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau MACDBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

```
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iMACD
//--- si FillArraysFromBuffers a rendu false, cela signifie que les données ne sont pa
   if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy)) return(0)
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur  Moving Averages Convergence/D
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iMACD       |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &macd_buffer[],    // le tampon d'indicateur des val
                           double &signal_buffer[],  // le tampon d'indicateur de la l
                           int ind_handle,           // le handle de l'indicateur iMAC
                           int amount                // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iMACDBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iMACD, le co
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau SignalBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iMACD, le co
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iOBV

Rend le handle de l'indicateur On Balance Volume. Seulement un tampon.

```
int  iOBV(
   string                 symbol,             // nom du symbole
   ENUM_TIMEFRAMES        period,             // période
   ENUM_APPLIED_VOLUME    applied_volume      // type de volume pour le calcul
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

applied_volume

[in] Le volume utilisé. Peut être chacune des valeurs de l'énumération ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOBV.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iOBV."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---  iOBV
#property indicator_label1  "iOBV"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOBV ,             // utiliser iOBV
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iOBV;               // le type de la fonction
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // le type du volume
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- les tampons d'indicateur
double         iOBVBuffer[];
//--- la variable pour stocker le handle de l'indicateur iOBV
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur On Balance Volume
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iOBV)
      handle=iOBV(name,period,applied_volume);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- le type du volume
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_OBV,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iOBV pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame On Balance Volume a été calculé
   short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iOBV
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
```

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iOBVBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iOBV
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- etiendrons le nombre de valeurs dans l'indicateur On Balance Volume
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iOBV          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &obv_buffer[],  // le tampon d'indicateur des valeurs
                         int ind_handle,        // le handle de l'indicateur iOBV
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iOBVBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iOBV, le cod
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
```

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iSAR

Rend le handle de l'indicateur Parabolic Stop and Reverse system. Seulement un tampon.

```
int  iSAR(
   string           symbol,       // nom du symbole
   ENUM_TIMEFRAMES  period,       // période
   double           step,         // le pas du surcroît de vitesse - l'accélération
   double           maximum       //le coefficient maximum du parcours après le prix
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

step

[in] L'augmentation de niveau de Stop est 0.02 d'habitude.

maximum

[in] Le niveau de Stop maximum est 0.2 d'habitude.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iSAR.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iSAR."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Parabolic Stop and Revers

#property indicator_chart_window
#property indicator_buffers 1
```

```
#property indicator_plots   1
//--- la construction iSAR
#property indicator_label1  "iSAR"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iSAR,              // utiliser iSAR
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iSAR;               // le type de la fonction
input double               step=0.02;                    // le pas - c'est le facteur
input double               maximum=0.2;                  // la valeur maximale du pas
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- les tampons d'indicateur
double         iSARBuffer[];
//--- la variable pour stocker le handle de l'indicateur iSAR
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Parabolic SAR
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
//--- définissons pour la propriété PLOT_ARROW le code du symbole de l'ensemble Wingdi
   PlotIndexSetInteger(0,PLOT_ARROW,159);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iSAR)
      handle=iSAR(name,period,step,maximum);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la valeur du pas
      pars[0].type=TYPE_DOUBLE;
      pars[0].double_value=step;
      //--- la valeur limite du pas, qui peut être utilisée aux calculs
      pars[1].type=TYPE_DOUBLE;
      pars[1].double_value=maximum;
      handle=IndicatorCreate(name,period,IND_SAR,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iSAR pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Parabolic SAR a ét
   short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
                           step,maximum);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

```
//--- le nombre de valeurs copiées de l'indicateur iSAR
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iSARBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iSAR
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur Parabolic SAR
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iSAR          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &sar_buffer[],  // le tampon d'indicateur des valeurs
                         int ind_handle,        // le handle de l'indicateur iSAR
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
```

```
//--- remplissons la partie du tableau iSARBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iSAR, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iRSI

Rend le handle de l'indicateur Relative Strength Index. Seulement un tampon.

```
int  iRSI(
   string               symbol,            // nom du symbole
   ENUM_TIMEFRAMES      period,            // période
   int                  ma_period,         // période de la prise de moyenne
   ENUM_APPLIED_PRICE   applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'index.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRSI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iRSI."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Relative Strength Index s

#property indicator_separate_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iRSI
#property indicator_label1  "iRSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les limites pour l'affichage des valeurs dans la fenêtre de l'indicateur
#property indicator_maximum 100
#property indicator_minimum 0
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  70.0
#property indicator_level2  30.0
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRSI,              // utiliser iRSI
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iRSI;               // le type de la fonction
input int                  ma_period=14;                 // la période de la prise de
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;    // le type du prix
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- le tampon d'indicateur
double         iRSIBuffer[];
//--- la variable pour stocker le handle de l'indicateur iRSI
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Relative Strength Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
```

© 2000-2024, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iRSI)
      handle=iRSI(name,period,ma_period,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- la valeur limite du pas, qui peut être utilisée aux calculs
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_RSI,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iRSI pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Relative Strength
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
                           ma_period,applied_price);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iRSI
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iRSIBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur iRSI
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//---retiendrons le nombre de valeurs dans l'indicateur Relative Strength Index
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iRSI          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rsi_buffer[],  // le tampon d'indicateur des valeurs
                         int ind_handle,        // le handle de l'indicateur iSAR
```

```
                          int amount              // le nombre de valeurs copiées
                          )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iRSIBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iRSI, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iRVI

Rend le handle de l'indicateur Relative Vigor Index.

```
int  iRVI(
   string           symbol,        // nom du symbole
   ENUM_TIMEFRAMES  period,        // période
   int              ma_period      // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'index.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRVI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iRVI."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Relative Vigor Index star

#property indicator_separate_window
#property indicator_buffers 2
```

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_plots   2
//--- la construction RVI
#property indicator_label1  "RVI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRVI,              // utiliser iRVI
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iRVI;               // le type de la fonction
input int                  ma_period=10;                 // la période des calculs
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- les tampons d'indicateur
double         RVIBuffer[];
double         SignalBuffer[];
//--- la variable pour stocker le handle de l'indicateur iRVI
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Relative Vigor Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
```

© 2000-2024, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iRVI)
      handle=iRVI(name,period,ma_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      //--- la période pour les calculs
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_RVI,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iRVI pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Relative Vigor Inc
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

```
  {
//--- le nombre de valeurs copiées de l'indicateur iRVI
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau RVIBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iRVI
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Relative Vigor Index
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iRVI        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rvi_buffer[],     // le tampon d'indicateur des valeu
                         double &signal_buffer[],  // le tampon d'indicateur de la lig
                         int ind_handle,           // le handle de l'indicateur iRVI
                         int amount                // le nombre de valeurs copiées
                         )
  {
```

```
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iRVIBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iRVI, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- remplissons la partie du tableau SignalBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iRVI, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iStdDev

Rend le handle de l'indicateur Standard Deviation. Seulement un tampon.

```
int  iStdDev(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 ma_period,         // période de la prise de moyenne
   int                 ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_MA_METHOD      ma_method,         // type de lissage
   ENUM_APPLIED_PRICE  applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom du symbole de l'instrument, dont les données doivent être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

ma_method

[in] La méthode de la prise de moyenne. Peut être l'une des valeurs ENUM_MA_METHOD.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iStdDev.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
```

```
#property description "des tampons d'indicateur pour l'indicateur technique iStdDev."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Standard Deviation standa

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iStdDev
#property indicator_label1  "iStdDev"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStdDev,           // utiliser iStdDev
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iStdDev;         // le type de la fonction
input int                  ma_period=20;              // la période de la prise de moy
input int                  ma_shift=0;                // le décalage
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // le type du lissage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iStdDevBuffer[];
//--- la variable pour stocker le handle de l'indicateur iStdDev
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Standard Deviation
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);
//--- spécifions le décalage
```

```
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iStdDev)
      handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du lissage
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iStdDev pour la
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Standard Deviatio
   short_name=StringFormat("iStdDev(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(ma_method),EnumToString(app
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iStdDev
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iStdDevBuffer est plus grand, que les valeurs dans l'indicat
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Standard Deviation
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
```

```
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Standard Deviation
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iStdDev       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &std_buffer[],  // le tampon d'indicateur de la ligne
                         int std_shift,         // le décalage de la ligne Standard De
                         int ind_handle,        // le handle de l'indicateur iStdDev
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iStdDevBuffer par les valeurs du tampon d'indic
   if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iStdDev, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iStochastic

Rend le handle de l'indicateur Stochastic Oscillator.

```
int  iStochastic(
   string           symbol,          // nom du symbole
   ENUM_TIMEFRAMES  period,          // période
   int              Kperiod,         // K-période (le nombre de barres pour le calcul)
   int              Dperiod,         // D-période (la période de premier lissage)
   int              slowing,         // lissage final
   ENUM_MA_METHOD   ma_method,       // type de lissage
   ENUM_STO_PRICE   price_field      // méthode de calcul stochastique
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

Kperiod

[in] La K-période (le nombre de barres) pour le calcul de la ligne %K.

Dperiod

[in] La période de la prise de moyenne pour le calcul de la ligne %D.

slowing

[in] La valeur du ralentissement.

ma_method

[in] La méthode de la prise de moyenne. Peut être l'une des valeurs ENUM_MA_METHOD.

price_field

[in] Le paramètre du choix des prix pour le calcul. Peut être une des valeurs ENUM_STO_PRICE.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                              Demo_iStochastic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
```

```
//|                                                  https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iStochasti
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Stochastic Oscillator sta

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- la construction Stochastic
#property indicator_label1  "Stochastic"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- spécifions les valeurs de frontière de l'indicateur
#property indicator_minimum 0
#property indicator_maximum 100
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStochastic,       // utiliser iStochastic
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iStochastic;     // le type de la fonction
input int                  Kperiod=5;                 // la période K (le nombre de ba
input int                  Dperiod=3;                 // la période D (la période du l
input int                  slowing=3;                 // la période du lissage définit
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // le type du lissage
input ENUM_STO_PRICE       price_field=STO_LOWHIGH;   // la méthode de calcul du stoch
input string               symbol=" ";                // le symbole
```

```
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- les tampons d'indicateur
double         StochasticBuffer[];
double         SignalBuffer[];
//--- la variable pour stocker le handle de l'indicateur iStochastic
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Stochastic Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iStochastic)
      handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[5];
      //--- la période K pour les calculs
      pars[0].type=TYPE_INT;
      pars[0].integer_value=Kperiod;
      //--- la période D pour le lissage primaire
      pars[1].type=TYPE_INT;
      pars[1].integer_value=Dperiod;
      //--- la période K pour le lissage définitif
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slowing;
      //--- le type du lissage
      pars[3].type=TYPE_INT;
```

© 2000-2024, MetaQuotes Ltd.

```
      pars[3].integer_value=ma_method;
      //--- la méthode de calcul du stochastique
      pars[4].type=TYPE_INT;
      pars[4].integer_value=price_field;
      handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iStochastic pou
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Stochastic Oscilla
   short_name=StringFormat("iStochastic(%s/%s, %d, %d, %d, %s, %s)",name,EnumToString
                           Kperiod,Dperiod,slowing,EnumToString(ma_method),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iStochastic
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
```

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau StochasticBuffer est plus grand, que les valeurs dans l'indi
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iStochastic
//--- si FillArraysFromBuffers a rendu false, cela signifie que les données ne sont pa
   if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy)) ret
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Stochastic Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iStochastic |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &main_buffer[],    // le tampon d'indicateur des val
                           double &signal_buffer[],  // le tampon d'indicateur de la l
                           int ind_handle,           // le handle de l'indicateur iSto
                           int amount                // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau StochasticBuffer par les valeurs du tampon d'in
   if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iStochastic,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- remplissons la partie du tableau SignalBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
```

```
      {
       //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
       PrintFormat("On n'a pas réussi à copier les données de l'indicateur iStochastic,
       //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
       return(false);
      }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iTEMA

Rend le handle de l'indicateur Triple Exponential Moving Average. Seulement un tampon.

```
int  iTEMA(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 ma_period,         // période de la prise de moyenne
   int                 ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_APPLIED_PRICE  applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iTEMA.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iTEMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
```

```
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Triple Exponential Moving

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iTEMA
#property indicator_label1  "iTEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTEMA,             // utiliser iTEMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iTEMA;           // le type de la fonction
input int                  ma_period=14;              // la période de la prise de moy
input int                  ma_shift=0;                // le décalage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iTEMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iTEMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Triple Exponential Moving Aver
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
//--- spécifions le décalage
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
```

© 2000-2024, MetaQuotes Ltd.

```
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iTEMA)
      handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[3];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du prix
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iTEMA pour la p
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Triple Exponential
   short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2024, MetaQuotes Ltd.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iTEMA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iTEMABuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Triple Exponential Moving
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- -retiendrons le nombre de valeurs dans l'indicateur Triple Exponential Moving Av
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Remplissons le tampon d'indicateur de l'indicateur iTEMA         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &tema_buffer[], // le tampon d'indicateur des valeurs
                         int t_shift,           // le décalage de la ligne
                         int ind_handle,        // le handle de l'indicateur iTEMA
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iTEMABuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iTEMA, le co
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iTriX

Rend le handle de l'indicateur Triple Exponential Moving Averages Oscillator. Seulement un tampon.

```
int  iTriX(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 ma_period,         // période de la prise de moyenne
   ENUM_APPLIED_PRICE  applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iTriX.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iTriX."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
```

```
#property indicator_plots   1
//--- la construction iTriX
#property indicator_label1  "iTriX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTriX,              // utiliser iTriX
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation              type=Call_iTriX;           // le type de la fonction
input int                   ma_period=14;              // la période
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // le type du prix
input string                symbol=" ";                // le symbole
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iTriXBuffer[];
//--- la variable pour stocker le handle de l'indicateur iTriX
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Triple Exponential Moving Aver
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- créons le handle de l'indicateur
   if(type==Call_iTriX)
      handle=iTriX(name,period,ma_period,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le type du prix
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iTriX pour la p
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Triple Exponential
   short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iTriX
   int values_to_copy;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iTriXBuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Triple Exponential Moving
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Triple Exponential Moving Ave
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iTriX         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &trix_buffer[], // le tampon d'indicateur des valeurs
                         int ind_handle,        // le handle de l'indicateur iTriX
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iTriXBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
       PrintFormat("On n'a pas réussi à copier les données de l'indicateur iTriX, le co
       //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
       return(false);
      }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iWPR

Rend le handle de l'indicateur Larry Williams' Percent Range. Seulement un tampon.

```
int  iWPR(
   string           symbol,          // nom du symbole
   ENUM_TIMEFRAMES  period,          // période
   int              calc_period      // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

calc_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iWPR.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iWPR."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iWPR
#property indicator_label1  "iWPR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrCyan
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- spécifions les valeurs de frontière de l'indicateur
#property indicator_minimum -100
#property indicator_maximum 0
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  -20.0
#property indicator_level2  -80.0
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iWPR,              // utiliser iWPR
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iWPR;            // le type de la fonction
input int                  calc_period=14;            // la période
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iWPRBuffer[];
//--- la variable pour stocker le handle de l'indicateur iWPR
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Larry Williams' Percent Range
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- créons le handle de l'indicateur
   if(type==Call_iWPR)
      handle=iWPR(name,period,calc_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=calc_period;
      handle=IndicatorCreate(name,period,IND_WPR,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iWPR pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Williams' Percent
   short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iWPR
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iWPRBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iATRBuffer  par les valeurs de l'indicateur Williams' Per
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Williams' Percent Range
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iWPR          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &wpr_buffer[], // le tampon d'indicateur des valeurs W
                         int ind_handle,        // le handle de l'indicateur iWPR
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iWPRBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iWPR, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
```

```
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iVIDyA

Rend le handle de l'indicateur Variable Index Dynamic Average. Seulement un tampon.

```
int  iVIDyA(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 cmo_period,        // période Chande Momentum
   int                 ema_period,        // période du facteur de lissage
   int                 ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_APPLIED_PRICE  applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

cmo_period

[in] La période (le nombre de barres) pour le calcul Chande Momentum Oscillator.

ema_period

[in] La période (le nombre de barres) EMA pour le calcul du facteur de lissage.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iVIDyA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
```

```
#property description "des tampons d'indicateur pour l'indicateur technique iVIDyA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Variable Index Dynamic Av

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iVIDyA
#property indicator_label1  "iVIDyA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVIDyA,            // utiliser iVIDyA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iVIDyA;          // le type de la fonction
input int                  cmo_period=15;             // la période Chande Momentum
input int                  ema_period=12;             // la période du facteur du liss
input int                  ma_shift=0;                // le décalage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iVIDyABuffer[];
//--- la variable pour stocker le handle de l'indicateur iVIDyA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Variable Index Dynamic Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);
//--- spécifions le décalage
```

```
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iVIDyA)
      handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période Chande Momentum
      pars[0].type=TYPE_INT;
      pars[0].integer_value=cmo_period;
      //--- la période du facteur du lissage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ema_period;
      //--- le décalage
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_shift;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iVIDyA pour la
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Triple Exponential
   short_name=StringFormat("iVIDyA(%s/%s, %d, %d, %d, %s)",name,EnumToString(period),
                           cmo_period,ema_period,ma_shift,EnumToString(applied_price))
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
```

© 2000-2024, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iVIDyA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iWPRBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Variable Index Dynamic Av
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
```

```
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Variable Index Dynamic Averag
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iVIDyA        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &vidya_buffer[],// le tampon d'indicateur des valeurs
                         int v_shift,           // le décalage de la ligne
                         int ind_handle,        // le handle de l'indicateur iVIDyA
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iVIDyABuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iVIDyA, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iVolumes

Rend le handle de l'indicateur reflétant les volumes. Seulement un tampon.

```
int  iVolumes(
   string                symbol,              // nom du symbole
   ENUM_TIMEFRAMES       period,              // période
   ENUM_APPLIED_VOLUME   applied_volume       // type de volume
   )
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

applied_volume

[in] Le volume utilisé. Peut être chacune des valeurs de l'énumération ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iVolumes.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iVolumes."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- la construction iVolumes
#property indicator_label1  "iVolumes"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVolumes,          // utiliser iVolumes
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iVolumes;           // le type de la fonction
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // le type du volume
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- les tampons d'indicateur
double         iVolumesBuffer[];
double         iVolumesColors[];
//--- la variable pour stocker le handle de l'indicateur iVolumes
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Volumes
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau aux tampons d'indicateur
   SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iVolumes)
      handle=iVolumes(name,period,applied_volume);
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- remplirons la structure par les valeurs des paramètres de l'indicateur
       MqlParam pars[1];
       //--- le type du prix
       pars[0].type=TYPE_INT;
       pars[0].integer_value=applied_volume;
       handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
      }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
      {
       //--- informons sur l'échec et déduisons le numéro de l'erreur
       PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iVolumes pour l
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- le travail de l'indicateur est terminé avant terme
       return(INIT_FAILED);
      }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Volumes a été calc
   short_name=StringFormat("iVolumes(%s/%s, %s)",name,EnumToString(period),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iVolumes
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
      {
       PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
       return(0);
      }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
```

```
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iVolumesBuffer r est plus grand, que les valeurs dans l'indi
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iVolumes
//--- si FillArraysFromBuffers a rendu false, cela signifie que les données ne sont pa
   if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy)) ret
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Volumes
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iVolumes    |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &volume_buffer[],   // le tampon d'indicateur des va
                           double &color_buffer[],    // le tampon d'indicateur des co
                           int ind_handle,            // le handle de l'indicateur iVo
                           int amount                 // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iVolumesBuffer par les valeurs du tampon d'indi
   if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iVolumes, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- remplissons la partie du tableau iVolumesColors par les valeurs du tampon d'indi
```

```
   if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iVolumes, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

Utilisation des résultats de l'optimisation

Fonctions pour l'organisation du traitement personnalisé des résultats de l'optimisation dans le Strategy Tester. Elles peuvent être appelées lors de l'optimisation des agents de test, ainsi que localement dans les Expert Advisor et les scripts.

Lorsque vous exécutez un Expert Advisor dans le testeur de stratégie, vous pouvez créer votre propre réseau de données sur la base des types simples ou des structures simples(elles ne contiennent pas de chaînes, des classes d'objets ou des objets de tableaux dynamiques). Cet ensemble de données peut être enregistré en utilisant la fonctionFrameAdd() dans une structure spéciale appelée un frame. Au cours de l'optimisation d'un Expert Advisor, chaque agent peut envoyer une série de frames au terminal. Toutes les frames reçues sont écrites dans le fichier *.MQD nommé comme l'Expert Advisor dans le répertoire_du_terminal/MQL5/Files/Tester. La réception d'un frame dans le terminal client d'un agent de test génère l'événement TesterPass.

Les frames peuvent être stockées dans la mémoire de l'ordinateur et dans un fichier avec le nom spécifié. Le langage MQL5 ne fixe aucune limitation sur le nombre de frames.

Limites de mémoire et d'espace disque dans le MQL5 Cloud Network

La limitation suivante s'applique aux optimisations exécutées dans le MQL5 Cloud Network : l'Expert Advisor ne doit pas écrire sur le disque plus de 4 Go d'informations ou utiliser plus de 4 Go de RAM . Si la limite est dépassée, l'agent du réseau ne pourra pas effectuer le calcul correctement et vous ne recevrez pas le résultat. Vous serez cependant facturé pour tout le temps passé sur les calculs.

Si vous avez besoin d'obtenir des informations à chaque passe d'optimisation, envoyez des images sans écrire sur le disque. Pour éviter d'utiliser les opérations sur les fichiers dans les Expert Advisors lors des calculs dans le MQL5 Cloud Network, vous pouvez utiliser la vérification suivante :

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

Fonction	Action
FrameFirst	Déplace le pointeur du cadre de lecture au début et remet à zéro le filtre précédemment défini.
FrameFilter	Définit le filtre de lecture du frame et déplace le pointeur au début
FrameNext	Lit une frame et déplace le pointeur vers la suivante

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
FrameInputs	Reçoit les paramètres d'entrée, sur lesquels la frame a été formée
FrameAdd	Ajoute une frame avec des données
ParameterGetRange	Reçoit les valeurs et le pas de changement d'une variable d'entrée lors l'optimisation d'un Expert Advisor dans le Strategy Tester
ParameterSetRange	Spécifie l'utilisation de la variable d'entrée lors de l'optimisation d'un Expert Advisor dans le Strategy Tester : la valeur, le pas du changement, les valeurs initiales et finales

Voir aussi

Statistiques de test , Propriétés d'un programme MQL5 éxecuté

© 2000-2024, MetaQuotes Ltd.

FrameFirst

Convertit l'indicateur de la lecture des trames de l'optimisation au début et oblitére un filtre établi.

```
bool  FrameFirst();
```

La valeur rendue

Rend true en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

FrameFilter

Définit le filtre de la lecture des trames et met l'indicateur au début

```
bool  FrameFilter(
   const string  name,          // nom public / label
   long          id             // id public
   );
```

La valeur rendue

Rend true en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

Si à titre du premier paramètre on transmet la chaîne vide, le filtre fonctionnera seulement selon un paramètre numérique, c'est-à-dire toutes les trames avec id indiqué seront examinées. Si la valeur du deuxième paramètre est égale à ULONG_MAX, seulement le filtre de texte fonctionne.

L'appel FrameFilter("", ULONG_MAX) est égal à l'appelFrameFirst(), c'est-à-dire est équivalent à l'absence du filtre.

© 2000-2024, MetaQuotes Ltd.

FrameNext

Lit la trame courant et déplace l'indicateur au suivant. Il y a 2 variantes de la fonction.

1. L'appel avec la réception d'une valeur numérique

```
bool  FrameNext(
   ulong&   pass,       // le numéro du passage dans l'optimisation, sur lequel la tran
   string&  name,       // nom public / label
   long&    id,         // id public
   double&  value       // valeur
   );
```

2. L'appel avec la réception de toutes les données de la trame

```
bool  FrameNext(
   ulong&   pass,       // le numéro du passage dans l'optimisation, sur lequel la tran
   string&  name,       // nom public / label
   long&    id,         // id public
   double&  value,      // valeur
   void&    data[]      // le tableau de n'importe quel type
   );
```

Paramètres

pass

[out] Le numéro du passage à l'optimisation dans le testeur des stratégies.

name

[out] Le nom de l'identificateur.

id

[out] La valeur de l'identificateur.

value

[out] Une valeur numérique unique.

data

[out] Un tableau de n'importe quel type.

La valeur rendue

Rend true en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

A l'utilisation de la deuxième variante de l'appel il est nécessaire de traiter correctement les données reçues au tableau*data[]*.

© 2000-2024, MetaQuotes Ltd.

FrameInputs

Reçoitles paramètres input, sur lesquels la trame avec le numéro spécifiée du passage a été formée.

```
bool  FrameInputs(
   ulong    pass,                // le numéro du passage dans l'optimisation
   string&  parameters[],        // le tableau de chaînes de l'aspect "parameterN=valu
   uint&    parameters_count     // le nombre total de paramètres
   );
```

Paramètres

pass

[in] Le numéro du passage à l'optimisation dans le testeur des stratégies.

parameters

[out] Un tableau de chaînes décrivant les noms et les valeurs des paramètres

parameters_count

[out] Le nombre d'éléments dans le tableau *parameters[]*.

La valeur rendue

Rend true en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

Ayant reçu la quantité de chaînes*parameters_count* dans le tableau*parameters[]*, on peut organiser le cycle pour le balayage de toutes les inscriptions. Cela permettra d'apprendre la valeur des paramètres d'entrée de l'expert pour le numéro spécifié du passage.

© 2000-2024, MetaQuotes Ltd.

FrameAdd

Ajoute la trame avec les données Il y a 2 variantes de la fonction.

1. L'ajout des données à partir d'un fichier

```
bool  FrameAdd(
   const string  name,         // nom public / label
   long          id,           // id public
   double        value,        // valeur
   const string  filename      // le nom du fichier avec les données
   );
```

2. L'ajout des données du tableau de n'importe quel type

```
bool  FrameAdd(
   const string  name,         // nom public / label
   long          id,           // id public
   double        value,        // valeur
   const void&   data[]        // le tableau de n'importe quel type
   );
```

Paramètres

name

[in] Le label public la trame. Peut être utilisé comme le filtre dans la fonction FrameFilter().

id

[in] L'identificateur public de la trame. Peut être utilisé comme le filtre dans la fonctionFrameFilter().

value

[in] La valeur numérique pour l'inscription dans la trame. Est destiné à la transmission du résultat unique du passage comme dans la fonction OnTester().

filename

[in] Le nom du fichier, qui contient des données pour l'ajout dans une trame. Le fichier doit être dans un dossierMQL5/Files.

data

[in] Le tableau de n'importe quel type pour l'inscription dans une trame est transmis selon la référence.

La valeur rendue

Rend true en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

ParameterGetRange

Reçoit pour la variable inputl'information sur la gamme des valeurs et sur le pas du changement à l'optimisation de l'expert dans le testeur des stratégies. Il y a 2 variantes de la fonction.

1. La réception de l'information pour le paramètre input du type entier

```
bool  ParameterGetRange(
   const string  name,          // le nom du paramètre (de la variable input)
   bool&         enable,        // l'optimisation du paramètre est autorisée
   long&         value,         // la valeur du paramètre
   long&         start,         //la valeur initiale
   long&         step,          // le pas du changement
   long&         stop           // la valeur finale
   );
```

2. La réception de l'information pour le paramètre input du type matériel

```
bool  ParameterGetRange(
   const string  name,          // le nom du paramètre (de la variable input)
   bool&         enable,        // l'optimisation du paramètre est autorisée
   double&       value,         // la valeur du paramètre
   double&       start,         // la valeur initiale
   double&       step,          // le pas du changement
   double&       stop           // la valeur finale
   );
```

Paramètres

name

[in] L'identificateur de la variable input. Telles variables sont les paramètres extérieurs du programme, dont les valeurs on peut spécifier au lancement sur le graphique ou au test séparé.

enable

[out] Le signe de ce qu'on peut utiliser ce paramètre pour le balayge des valeurs en train de l'optimisation dans le testeur des stratégies.

value

[out] La valeur du paramètre.

start

[out] La valeur initiale du paramètre à l'optimisation.

step

[out] Le pas du changement du paramètre au balayage de ses valeurs.

stop

[out] La valeur finale du paramètre à l'optimisation.

La valeur rendue

© 2000-2024, MetaQuotes Ltd.

Rend true en cas de l'exécution réussie, autrement - false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

La fonction peut être appelée seulement à partir de gestionnairesOnTesterInit(), OnTesterPass() etOnTesterDeinit(). Est destinée à la réception de la valeur et de la gamme du changement des paramètres d'entrée de l'expert en train de l'optimisation dans le testeur des stratégies.

A l'appel dans OnTesterInit() on peut utiliser l'information reçue pour la redéfinition de la règle du balayage de chaquevariable input à l'aide de la fonctionParameterSetRange(). Ainsi spécifier les nouvelles valeurs Start, Stop, Step et même exclure entièrement ce paramètre de l'optimisation malgré les réglages dans le testeur des stratégies. Cela permet de créer les scripts personnels pour la gestion de l'espace des paramètres d'entrée à l'optimisation, c'est-à-dire exclure de l'optimisation des paramètres en fonction des valeurs des paramètres clés de l'expert.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
#property description "L'expert pour la démonstration de la fonction ParameterGetRange
#property description "Il est nécessaire de lancer dans le testeur des stratégies en r
//--- input parameters
input int                  Input1=1;
input double               Input2=2.0;
input bool                 Input3=false;
input ENUM_DAY_OF_WEEK     Input4=SUNDAY;

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- l'expert est destiné seulement au travail dans le testeur des stratégies
   if(!MQL5InfoInteger(MQL5_OPTIMIZATION))
     {
      MessageBox("Il est nécessaire de lancer dans le testeur des stratégies en mode o
      //---terminons le travail de l'expert avant terme et retirons du graphique
      return(INIT_FAILED);
     }
//---l'achèvement réussi de l'initialisation
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| TesterInit function                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- l'exemple pour le paramètre input du type long
   string name="Input1";
   bool enable;
   long par1,par1_start,par1_step,par1_stop;
   ParameterGetRange(name,enable,par1,par1_start,par1_step,par1_stop);
   Print("Le premier paramètre");
   PrintFormat("%s=%d  enable=%s  from %d to %d with step=%d",
               name,par1,(string)enable,par1_start,par1_stop,par1_step);
//--- l'exemple pour le paramètre input du type double
   name="Input2";
   double par2,par2_start,par2_step,par2_stop;
   ParameterGetRange(name,enable,par2,par2_start,par2_step,par2_stop);
   Print("Le deuxième paramètre le paramètre");
   PrintFormat("%s=%G  enable=%s  from %G to %G with step=%G",
               name,par2,(string)enable,par2_start,par2_stop,par2_step);

//--- l'exemple pour le paramètre input du type bool
   name="Input3";
   long par3,par3_start,par3_step,par3_stop;
   ParameterGetRange(name,enable,par3,par3_start,par3_step,par3_stop);
   Print("Le troisième paramètre");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,(string)par3,(string)enable,
               (string)par3_start,(string)par3_stop);
//--- l'exemple pour le paramètre input du type l'énumération
   name="Input4";
   long par4,par4_start,par4_step,par4_stop;
   ParameterGetRange(name,enable,par4,par4_start,par4_step,par4_stop);
   Print("Le quatrième paramètre");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,EnumToString((ENUM_DAY_OF_WEEK)par4),(string)enable,
               EnumToString((ENUM_DAY_OF_WEEK)par4_start),
               EnumToString((ENUM_DAY_OF_WEEK)par4_stop));
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| TesterDeinit function                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- ce message sortira à la fin de l'optimisation
   Print(__FUNCTION__," Optimisation completed");
  }
```

© 2000-2024, MetaQuotes Ltd.

ParameterSetRange

Établit des règles d'utilisation de la variable d'entrée lors de l'optimisation d'un Expert Advisor dans le Strategy Tester : la valeur, le pas du changement, les valeurs initiales et finales. Il existe 2 variantes de la fonction.

1. Définition des valeurs pour le paramètre d'entrée de type entier

```
bool  ParameterSetRange(
   const string  name,          // le nom du paramètre (de la variable d'entrée)
   bool          enable,        // autoriser l'optimisation du paramètre
   long          value,         // la valeur du paramètre du démarrage long
   long          start,         // la valeur initiale
   long          step,          // le pas du changement
   long          stop           // la valeur finale
   );
```

2. Définition des valeurs pour le paramètre d'entrée de type réel

```
bool  ParameterSetRange(
   const string  name,          // le nom du paramètre (de la variable d'entrée)
   bool          enable,        // autoriser l'optimisation du paramètre
   double        value,         // la valeur du paramètre
   double        start,         // la valeur initiale
   double        step,          // le pas du changement
   double        stop           // la valeur finale
   );
```

Paramètres

name

[in] L'identifiant de la variable input ou sinput. Ces variables sont des paramètres externes d'une application. Leurs valeurs peuvent être spécifiées lors du lancement du programme.

enable

[in] Activer ce paramètre pour énumérer les valeurs lors de l'optimisation dans le Strategy Tester.

value

[in] La valeur du paramètre.

start

[in] La valeur initiale du paramètre lors de l'optimisation.

step

[in] Le pas du changement du paramètre lors de l'énumération de ses valeurs.

stop

[in] La valeur finale du paramètre à l'optimisation.

Valeur de retour

© 2000-2024, MetaQuotes Ltd.

Retourne true en cas d'exécution réussie - false sinon. Pour obtenir des informations sur l'erreur, utilisez la fonction GetLastError().

Note

La fonction peut être appelée seulement depuis le gestionnaire OnTesterInit() lors du lancement de l'optimisation dans le Strategy Tester. Elle est conçue pour spécifier la plage et le pas du changement du paramètre. Le paramètre peut être complètement exclu de l'optimisation indépendamment des paramètres du Strategy Tester. Il permet également d'utiliser les variables d'entrée déclarées dans le processus d'optimisation.

La fonction ParameterSetRange() vous permet de gérer une optimisation d'un Expert Advisor dans le Strategy Tester en fonction des valeurs de ses paramètres clés en incluant ou en excluant les paramètres d'entrée requis à partir de l'optimisation, du réglage de la plage requise et du pas du changement.

© 2000-2024, MetaQuotes Ltd.

Le travail avec les événements

Les fonctions pour le travail avec les événements d'utilisateur et les événements du minuteur. Sauf ces fonctions il y a des fonctions spéciales pour le traitement des événements prédéterminés.

Fonction	Action
EventSetMillisecondTimer	Lance le générateur des événements de la minuterie d'une haute permission avec la période moins 1 seconde pour un graphique courant
EventSetTimer	Lance le générateur des événements du minuteur avec la périodicité indiquée pour le graphique courant
EventKillTimer	Arrête sur le graphique courant la génération des événements selon le minuteur
EventChartCustom	Génère l'événement d'utilisateur pour le graphique indiqué

Voir aussi

Les types des événements du graphique

© 2000-2024, MetaQuotes Ltd.

EventSetMillisecondTimer

Indique au terminal de client que pour l'expert donné ou l'indicateur il est nécessaire de générer les événements dela minuterieavec la périodicité moins d'une seconde.

```
bool  EventSetMillisecondTimer(
   int  milliseconds      // le nombre de millisecondes
   );
```

Réglages

milliseconds

[in] Le nombre de millisecondes définissant la périodicité de l'apparition des événements de la minuterie.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement rend false. Pour recevoir le codede l'erreur il faut appeler la fonction GetLastError().

Note

Cette fonction est destinée aux cas, quand il fautla minuterie d'une haute permission, c'est-à-dire il faut recevoir les événements de la minuterie plus souvent, qu'une fois par seconde. Si la minuterie simple avec une période plus de 1 seconde est assez pour vous, utilisezEventSetTimer().

Dans le cas total à la réduction de la période de la minuterie augmente le temps du test, puisque augmente le nombre d'appels du gestionnaire des événements de la minuterie. Au travail en mode du temps réel les événements de la minuterie sont générés non plus souvent que 1 fois à 10-16 millisecondes ce qui est dû aux contraintes matérielles.

D'habitude, cette fonction doit être appelée de la fonctionOnInit() ou dans le constructeurde la classe. Pour traiter les événements provenant de la minuterie, l'expert ou l'indicateur doit avoir la fonctionOnTimer().

Chaque expert et chaque indicateur travaille avec sa minuterie et reçoit les événements seulement de lui. A l'achèvement du travail du programme mql5 la minuterie est supprimée forcément, si elle était créé, mais n'était pas déconnectée par la fonction EventKillTimer().

Pour chaque programme peut être lancé pas plus d'une minuterie. Chaque programme MQL5 et chaque graphique a sa propre file d'attente d'événements, où sont ajoutés tous les nouveaux événements entrants. Si l'événementTimerse trouve déjà dans la file d'attente ou cet événement est en état du traitement, on ne met pas un nouvel événement Timer dans la file d'attente du programme mql5.

© 2000-2024, MetaQuotes Ltd.

EventSetTimer

Indique au terminal de client que pour l'expert donné ou l'indicateur il est nécessaire de générer les événements de minuteur avec la périodicité indiquée.

```
bool  EventSetTimer(
   int  seconds      // nombre de secondes
   );
```

Paramètres

seconds

[in] Le nombre de secondes définissant la périodicité de l'apparition des événements de minuteur.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false.Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Note

D'habitude, cette fonction doit être appelé de la fonction OnInit() ou dans le constructeur de la classe. Pour traiter les événements venant de minuteur, l'expert ou l'indicateur doit avoir la fonction OnTimer().

Chaque expert et chaque indicateur travaille avec le minuteur et reçoit les événements seulement de lui. A l'achèvement du travail du programme mql5 le minuteur est supprimé forcément, s'il était créé, mais n'était pas débrancher par la fonction EventKillTimer().

Pour chaque programme pas plus qu'un minuteur peut être lancé. Chaque programme MQL5 et chaque graphique a sa propre file d'attente d'événements, où sont ajoutés tous les nouveaux événements entrants. Si l'événementTimerse trouve déjà dans la file d'attente ou cet événement est en état du traitement, on ne met pas un nouvel événement Timer dans la file d'attente du programme mql5.

© 2000-2024, MetaQuotes Ltd.

EventKillTimer

Indique au terminal de client qu'il est nécessaire d'arrêter la génération des événements de minuteur pour l'expert donné ou l'indicateur.

```
void  EventKillTimer();
```

La valeur rendue

Il n'y a pas de valeur rendue.

Note

D'habitude, cette fonction doit être appelé de la fonction OnDeinit() dans le cas où en fonction OnInit() la fonction EventSetTimer() a été appelée. Ou doit être appelé du destructeur de la classe, si dans le constructeur de cette classe la fonction EventSetTimer() a été appelée.

Chaque expert et chaque indicateur travaille avec le minuteur et reçoit les événements seulement de lui. A l'achèvement du travail du programme mql5 le minuteur est supprimé forcément, s'il était créé, mais n'était pas débrancher par la fonction EventKillTimer().

© 2000-2024, MetaQuotes Ltd.

EventChartCustom

Génère l'événement d'utilisateur pour le graphique indiqué.

```
bool  EventChartCustom(
   long    chart_id,            // identificateur du graphique-destinataire de l'événe
   ushort  custom_event_id,     // identificateur de l'événement
   long    lparam,              // paramètre du type long
   double  dparam,              // paramètre du type double
   string  sparam               // paramètre de chaîne de l'événement
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

custom_event_id

[in] L'identificateur de l'événement d'utilisateur. Cet identificateur est ajouté automatiquement à la valeur CHARTEVENT_CUSTOM et est transcrit au type entier.

lparam

[in] Le paramètre de l'événement du type long, transmis à la fonction OnChartEvent.

dparam

[in] Le paramètre de l'événement du type double, transmis à la fonction OnChartEvent.

sparam

[in] Le paramètre de l'événement du type string, transmis à la fonction OnChartEvent. Si la chaîne a la longueur plus que 63 symboles, la chaîne est tronquée

La valeur rendue

Rend true en cas de la position réussie de l'événement d'utilisateur au tour des événements du graphique - du destinataire de l'événement. En cas de l'erreur rend false, pour recevoir un code de l'erreur utilisez GetLastError().

Note

L'expert ou l'indicateur fixé au graphique indiqué, traite l'événement donné à l'aide de la fonction OnChartEvent(int event_id, long& lparam, double& dparam, string& sparam).

Pour chaque type de l'événement les paramètres d'entrée de la fonction OnChartEvent() ont les valeurs définies, qui sont nécessaires au traitement de cet événement. Dans le tableau sont énumérés les événements et les valeurs, qui sont transmises par les paramètres.

L'événement	La valeur du paramètre id	La valeur du paramètre lparam	La valeur du paramètre dparam	La valeur du paramètre sparam
L'événement de la pression du clavier	CHARTEVENT_KEYDOWN	le code de la touche appuyée	Le nombre de pressions du bouton générées	La valeur de ligne d'un masque de bit,

© 2000-2024, MetaQuotes Ltd.

L'événement	La valeur du paramètre id	La valeur du paramètre lparam	La valeur du paramètre dparam	La valeur du paramètre sparam
			au cours de sa rétention dans l'état appuyé	décrivant le statut des boutons du clavier
Les événements de la souris(si la propriété est définie pour le graphique CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	La coordonnée X	La coordonnée Y	La valeur de chaîne d'un masque de bits décrivant l'état de boutons de souris
L'événement de la création de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_CREATE=true)	CHARTEVENT_OBJECT_CREATE	—	—	Le nom de l'objet créé graphique
L'événement du changement des propriétés de l'objet par le dialogue des propriétés	CHARTEVENT_OBJECT_CHANGE	—	—	Le nom de l'objet créé graphique
L'événement de la suppression de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE=true)	CHARTEVENT_OBJECT_DELETE	—	—	Le nom de l'objet graphique supprimé
L'événement du clic de la souris sur le graphique	CHARTEVENT_CLICK	La coordonnée X	La coordonnée Y	—
L'événement du clic de la souris sur l'objet graphique	CHARTEVENT_OBJECT_CLICK	La coordonnée X	La coordonnée Y	Le nom d'un objet graphique sur lequel

© 2000-2024, MetaQuotes Ltd.

L'événement	La valeur du paramètre id	La valeur du paramètre lparam	La valeur du paramètre dparam	La valeur du paramètre sparam
				l'événement s'est produit
L'événement du déplacement de l'objet graphique à l'aide de la souris	CHARTEVENT_OBJECT_DRAG	—	—	Le nom de l'objet graphique déplacé
L'événement de la fin de l'édition du texte dans l'objet graphique "Le champ de l'entrée"	CHARTEVENT_OBJECT_ENDEDIT	—	—	Le nom de l'objet graphique "Le champ de l'entrée", où l'édition du texte s'est achevée
L'événement du changement du graphique	CHARTEVENT_CHART_CHANGE	—	—	—
L'événement d'utilisateur avec le numéro N	CHARTEVENT_CUSTOM+N	La valeur, spécifiée par la fonction EventChartCustom()	La valeur, spécifiée par la fonction EventChartCustom()	La valeur, spécifiée par la fonction EventChartCustom()

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          ButtonClickExpert.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

string buttonID="Button";
string labelID="Info";
int broadcastEventID=5000;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- créons le bouton, pour la transmission des événements d'utilisateur
   ObjectCreate(0,buttonID,OBJ_BUTTON,0,100,100);
   ObjectSetInteger(0,buttonID,OBJPROP_COLOR,clrWhite);
   ObjectSetInteger(0,buttonID,OBJPROP_BGCOLOR,clrGray);
   ObjectSetInteger(0,buttonID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_XSIZE,200);
   ObjectSetInteger(0,buttonID,OBJPROP_YSIZE,50);
   ObjectSetString(0,buttonID,OBJPROP_FONT,"Arial");
   ObjectSetString(0,buttonID,OBJPROP_TEXT,"Bouton");
   ObjectSetInteger(0,buttonID,OBJPROP_FONTSIZE,10);
   ObjectSetInteger(0,buttonID,OBJPROP_SELECTABLE,0);

//--- créons la marque pour afficher l'information
   ObjectCreate(0,labelID,OBJ_LABEL,0,100,100);
   ObjectSetInteger(0,labelID,OBJPROP_COLOR,clrRed);
   ObjectSetInteger(0,labelID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,labelID,OBJPROP_YDISTANCE,50);
   ObjectSetString(0,labelID,OBJPROP_FONT,"Trebuchet MS");
   ObjectSetString(0,labelID,OBJPROP_TEXT,"Pas d'information");
   ObjectSetInteger(0,labelID,OBJPROP_FONTSIZE,20);
   ObjectSetInteger(0,labelID,OBJPROP_SELECTABLE,0);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ObjectDelete(0,buttonID);
   ObjectDelete(0,labelID);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
```

```
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- contrôlerons l'événement sur la pression du bouton de la souris
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      string clickedChartObject=sparam;
      //--- si la pression sur l'objet avec le nom buttonID
      if(clickedChartObject==buttonID)
        {
         //--- Etat du bouton - on a appuyé le bouton ou non
         bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
         //--- notez un message de réglage
         Print("Bouton est appuyé = ",selected);
         int customEventID; // numéro de l'événement d'utilisateur pour l'expédition
         string message;    // message pour l'expédition dans l'événement
         //--- si le bouton est appuyé
         if(selected)
           {
            message="bouton est appuyé";
            customEventID=CHARTEVENT_CUSTOM+1;
           }
         else // bouton n'est pas appuyé
           {
            message="Bouton est appuyé";
            customEventID=CHARTEVENT_CUSTOM+999;
           }
         //--- envoyons l'événement d'utilisateur au graphique
         EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
         //--- envoyons le message aux tous les graphiques ouverts
         BroadcastEvent(ChartID(),0,"Broadcast Message");
         //--- message de réglage
         Print("On a expédié l'événement avec ID = ",customEventID);
        }
      ChartRedraw();// copions forcément tous les objets sur le graphique
     }

//--- vérifions l'événement sur l'appartenance aux événements d'utilisateur
   if(id>CHARTEVENT_CUSTOM)
     {
      if(id==broadcastEventID)
        {
         Print("Ont reçu le message de diffusion du graphique avec id = "+lparam);
        }
      else
        {
         //--- lirons le message de texte dans l'événement
         string info=sparam;
         Print("On traite l'événement D'UTILISATEUR avec  ID = ",id);
         //--- déduisons le message dans la marque
         ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
         ChartRedraw();// copions forcément tous les objets sur le graphique
        }
     }
  }
//+------------------------------------------------------------------+
//| envoyer le message de diffusion aux graphiques ouverts           |
//+------------------------------------------------------------------+
void BroadcastEvent(long lparam,double dparam,string sparam)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
   long currChart=ChartFirst();
   int i=0;
   while(i<CHARTS_MAX)                // nous avons sans faute plus de CHARTS_MAX des
     {
      EventChartCustom(currChart,eventID,lparam,dparam,sparam);
      currChart=ChartNext(currChart); // sur la base du précédent nous recevrons un no
      if(currChart==-1) break;        // ont atteint la fin de la liste des graphiques
      i++;                            // n'oublierons pas d'augmenter le compteur
     }
  }
//+------------------------------------------------------------------+
```

Voir aussi

Les événements du terminal de client, la Fonction du traitement des événements

© 2000-2024, MetaQuotes Ltd.

Travailler avec OpenCL

Les programmes OpenCL qui sont utilisés pour effectuer des calculs sur les cartes vidéo prenant en charge OpenCL 1.1 ou supérieur. Les cartes vidéo modernes contiennent des centaines de petits processeurs spécialisés qui peuvent effectuer simultanément des opérations mathématiques simples avec des flux de données entrants. Le langage OpenCL prend en charge l'organisation de tels calculs parallèles et offre une plus grande vitesse pour une certaine classe de tâches.

Dans certaines cartes graphiques, travailler avec des données de type double est désactivé par défaut. Cela peut conduire à des erreurs de compilation 5105. Pour activer le support pour les données de type double, veuillez ajouter la directive suivante à votre programme OpenCL : #pragma OPENCL EXTENSION cl_khr_fp64 : enable. Toutefois, si la carte graphique ne prend pas en charge les doubles, l'activation de cette directive ne sera d'aucun effet.

Il est recommandé d'écrire le code source OpenCL dans des fichiers CL séparés, pouvant être par la suite inclus dans le programme MQL5 en utilisant des variables de ressource.

Gestion des erreurs dans les programmes OpenCL

Pour obtenir des informations sur la dernière erreur dans un programme OpenCL, utilisez les fonctionsCLGetInfoInteger et CLGetInfoString qui permettent l'obtention du code d'erreur et de la description textuelle.

Dernier code d'erreur OpenCL : Pour obtenir la dernière erreur OpenCL, appelez la fonction CLGetInfoInteger, le paramètre *handle*est ignoré (peut être égal à 0). Description des erreurs :https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS.

Pour un code d'erreur inconnu, la chaîne "unknown OpenCL error N" () est retournée, où N est le code d'erreur. Exemple :

```
//--- le premier paramètre 'handle' est ignoré lorsque l'on récupère le dernier code d
intcode= (int)CLGetInfoInteger(0,CL_LAST_ERROR);
```

Description textuelle de l'erreur OpenCL : pour obtenir la dernière erreur OpenCL, appelez la fonction CLGetInfoString. Le code d'erreur doit être passé dans le paramètre*handle*.

Description des erreurs : https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Si CL_LAST_ERROR est passé à la place du code d'erreur, la fonction renvoie la description de la dernière erreur. Par exemple :

```
//--- récupère le code de la dernière erreur OpenCL
int   code= (int)CLGetInfoInteger(0,CL_LAST_ERROR);
stringdesc;// pour récupérer la description textuelle de l'erreur

//--- utilise le code d'erreur pour obtenir une description du texte de l'erreur
if(!CLGetInfoString(code,CL_ERROR_DESCRIPTION,desc))
 desc="impossible d'obtenir la description de l'erreur OpenCL, "+ (string)GetLastErro
Print(desc);

//--- pour obtenir la description de la dernière erreur OpenCL sans obtenir le code er
if(!CLGetInfoString(CL_LAST_ERROR,CL_ERROR_DESCRIPTION,desc))
 desc="impossible d'obtenir la description de l'erreur OpenCL, "+ (string)GetLastErro
```

© 2000-2024, MetaQuotes Ltd.

```
Print(desc);;
```

Pour l'instant, le nom dans l'énumération interne est donné comme description de l'erreur. Vous pouvez trouver sa description ici :https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Par exemple, la valeur CL_INVALID_KERNEL_ARGS signifie "Returned when enqueuing a kernel when some kernel arguments have not been set or are invalid" ("Renvoyé lors de la mise en file d'attente d'un noyau lorsque certains arguments du noyau n'ont pas été définis ou sont invalides.")

Fonctions pour l'exécution des programmes en OpenCL :

Fonction	Action
CLHandleType	Retourne le type du handle OpenCL comme une valeur de l'énumération ENUM_OPENCL_HANDLE_TYPE
CLGetInfoInteger	Retourne la valeur d'une propriété de type integer pour un objet ou un dispositif OpenCL
CLContextCreate	Crée un contexte OpenCL
CLContextFree	Supprime un contexte OpenCL
CLGetDeviceInfo	Reçoit la propriété du dispositif du pilote informatique OpenCL
CLProgramCreate	Crée un programme OpenCL à partir du code source
CLProgramFree	Supprime un programme OpenCL
CLKernelCreate	Crée une fonction de démarrage OpenCL
CLKernelFree	Supprime une fonction au démarrage de l'OpenCL
CLSetKernelArg	Définit un paramètre pour la fonction OpenCL
CLSetKernelArgMem	Définit un buffer OpenCL en tant que paramètre de la fonction OpenCL
CLSetKernelArgMemLocal	Définit le buffer local comme un argument de la fonction du noyau
CLBufferCreate	Crée un buffer OpenCL
CLBufferFree	Supprime un buffer OpenCL
CLBufferWrite	Ecrit un tableau dans un buffer OpenCL
CLBufferRead	Lit un buffer OpenCL dans un tableau
CLExecute	Exécute un programme OpenCL
CLExecute	Retourne le statut d'exécution du programme OpenCL

Voir aussi

OpenCL, Ressources

© 2000-2024, MetaQuotes Ltd.

CLHandleType

Rend le type OpenCL du handle à titre de la valeur de l'énumération ENUM_OPENCL_HANDLE_TYPE.

```
ENUM_OPENCL_HANDLE_TYPE  CLHandleType(
   int  handle     // le handle de l'objet OpenCL
   );
```

Paramètres

handle

[in] Le handle sur l'objet OpenCL: le contexte, le noyau, le tampon ou le programme OpenCL.

La valeur rendue

Le type du handle OpenCL à titre de la valeur de l'énumération ENUM_OPENCL_HANDLE_TYPE.

ENUM_OPENCL_HANDLE_TYPE

Identificateur	La description
OPENCL_INVALID	Le handle incorrect
OPENCL_CONTEXT	Le handle du cotexte OpenCL
OPENCL_PROGRAM	Le handle du programme OpenCL
OPENCL_KERNEL	Le handle du noyau OpenCL
OPENCL_BUFFER	Le handle du tampon OpenCL

© 2000-2024, MetaQuotes Ltd.

CLGetInfoInteger

Retourne la valeur d'une propriété de type integer pour un objet ou un dispositif OpenCL

```
long  CLGetInfoInteger(
   int  handle,                          // le handler sur l'objet OpenCL ou le numé
   ENUM_OPENCL_PROPERTY_INTEGER  prop    // la propriété demandée
   );
```

Paramètres

handle

[in] Le handler sur l'objet OpenCL ou le numéro du dispositif OpenCL. Le numérotation des dispositifs OpenCL commence à zéro

prop

[in] L'identifiant de la propriété demandée dans l'énumération ENUM_OPENCL_PROPERTY_INTEGER et pour laquelle vous souhaitez obtenir la valeur.

Valeur de retour

La valeur de la propriété en cas de succès ou -1 en cas d'erreur Pour obtenir des informations sur l'erreur, utilisez la fonction GetLastError().

ENUM_OPENCL_PROPERTY_INTEGER

Identifiant	Description	Type
CL_DEVICE_COUNT	Le nombre de dispositifs avec le support d'OpenCL. Cette propriété ne nécessite pas de spécifier le premier paramètre, à savoir que vous pouvez passer une valeur nulle pour le paramètre *handle*.	int
CL_DEVICE_TYPE	Le type du dispositif	ENUM_CL_DEVICE_TYPE
CL_DEVICE_VENDOR_ID	L'identifiant unique du vendeur	uint
CL_DEVICE_MAX_COMPUTE_UNITS	Le nombre de tâches calculées parallèlement sur le périphérique OpenCL. Un groupe de travail travaille sur une tâche de calcul. La valeur minimale est de 1	uint
CL_DEVICE_MAX_CLOCK_FREQUENCY	La fréquence maximale en MHz du périphérique .	uint
CL_DEVICE_GLOBAL_MEM_SIZE	La taille de la mémoire globale du dispositif en octets	ulong

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description	Type
CL_DEVICE_LOCAL_MEM_SIZE	La taille de la mémoire locale des données traitées en octets	uint
CL_BUFFER_SIZE	Taille actuelle du buffer OpenCL en octets	ulong
CL_DEVICE_MAX_WORK_GROUP_SIZE	Le nombre total de groupe de travail locaux disponibles pour le périphérique OpenCL.	ulong
CL_KERNEL_WORK_GROUP_SIZE	Le nombre total de groupe de travail locaux disponibles pour le programme OpenCL.	ulong
CL_KERNEL_LOCAL_MEM_SIZE	La taille de la mémoire locale (en octets) utilisée par un programme OpenCL pour résoudre toutes les taches parallèles dans un groupe. Utilisez CL_DEVICE_LOCAL_MEM_SIZE pour récupérer la valeur disponible maximum	ulong
CL_KERNEL_PRIVATE_MEM_SIZE	La taille minimum de la mémoire privée (en octets) utilisée par chaque tâche du noyau du programme OpenCL.	ulong
CL_LAST_ERROR	La valeur de la dernière erreur OpenCL	int

L'énumération ENUM_CL_DEVICE_TYPE contient les types possibles des périphériques supportant OpenCL. Vous pouvez récupérer le type du périphérique par son numéro ou le handle de l'objet OpenCL à l'aide de l'appel CLGetInfoInteger(handle_or_deviceN, CL_DEVICE_TYPE).

ENUM_CL_DEVICE_TYPE

Identifiant	Description
CL_DEVICE_ACCELERATOR	Accélérateur OpenCL dédiés (par exemple l'IBM CELL Blade).
CL_DEVICE_CPU	Le périphérique OpenCL qui est le processeur hôte. Le processeur hôte exécute les implémentations OpenCL et est mono ou multi coeurs.
CL_DEVICE_GPU	Le dispositif OpenCL basé sur la carte vidéo.
CL_DEVICE_DEFAULT	Le dispositif OpenCL par défaut. Le dispositif CL_DEVICE_TYPE_CUSTOM ne peut pas être le

© 2000-2024, MetaQuotes Ltd.

Identifiant	Description
	dispositif par défaut.
CL_DEVICE_CUSTOM	Les accélérateurs spécialisés ne supportent pas les programmes écrits en OpenCL C.

Exemple :

```
void OnStart()
  {
   int cl_ctx;
//--- initialisation du contexte OpenCL
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL not found");
      return;
     }
//--- information totale sur le dispositif OpenCL
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY)," MHz"
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE)," by
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE)," byte
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

© 2000-2024, MetaQuotes Ltd.

CLGetInfoString

Rend la valeur de ligne pour l'objet OpenCL ou le dispositif.

```
bool  CLGetInfoString(
   int  handle,                          // le handle de l'objet OpenCL ou le numéro
   ENUM_OPENCL_PROPERTY_STRING  prop,    // la propriété demandée
   string&  value                        // la ligne selon la référence
   );
```

Réglages

handle

[in] Le handle sur l'objet OpenCL ou le numéro du dispositif OpenCL. Le numérotage OpenCL des dispositifs commence par le zéro.

prop

[in] Le type de la propriété demandée de l'énumération ENUM_OPENCL_PROPERTY_STRING, dont la valeur il faut recevoir.

value

[out] La ligne pour la réception de la valeur de la propriété.

La valeur rendue

En cas de l'exécution successive rend true, autrement rend false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

ENUM_OPENCL_PROPERTY_STRING

Identificateur	La description
CL_PLATFORM_PROFILE	CL_PLATFORM_PROFILE - Le profil OpenCL. Le nom du profil peut être une des valeurs: • FULL_PROFILE - la réalisation soutient OpenCL (la fonctionnalité est définie comme la partie de la spécification du noyau et ne demande pas les extensions supplémentaires pour le soutien d' OpenCL); • EMBEDDED_PROFILE - la réalisation soutient OpenCL en forme du complément. Le profil complété est défini comme la sous-multitude pour chaque version OpenCL.
CL_PLATFORM_VERSION	La version OpenCL
CL_PLATFORM_VENDOR	Le nom du fabricant de l'appareil
CL_PLATFORM_EXTENSIONS	La liste des extensions soutenus par la plateforme. Les noms des extensions doivent être soutenus par toutes les dispositifs liés avec cette plateforme
CL_DEVICE_NAME	Le nom du dispositif

© 2000-2024, MetaQuotes Ltd.

Identificateur	La description
CL_DEVICE_VENDOR	Le nom du fabricant
CL_DRIVER_VERSION	La version du pilote informatique OpenCL dans le format major_number.minor_number
CL_DEVICE_PROFILE	Le profil du dispositif OpenCL. Le nom du profil peut être une des valeurs: • FULL_PROFILE - la réalisation soutient OpenCL (la fonctionnalité est définie comme la partie de la spécification du noyau et ne demande pas les extensions supplémentaires pour le soutien d' OpenCL); • EMBEDDED_PROFILE - la réalisation soutient OpenCL en forme du complément. Le profil complété est défini comme la sous-multitude pour chaque version OpenCL.
CL_DEVICE_VERSION	La version OpenCL dans le format "OpenCL<space><major_version.minor_version><space><vendor-specific information>"
CL_DEVICE_EXTENSIONS	La liste des extensions soutenus par le dispositif. La liste peut insérer les extensions soutenus par le producteur et contenir un ou plus de noms approuvés: cl_khr_int64_base_atomics cl_khr_int64_extended_atomics cl_khr_fp16 cl_khr_gl_sharing cl_khr_gl_event cl_khr_d3d10_sharing cl_khr_dx9_media_sharing cl_khr_d3d11_sharing
CL_DEVICE_BUILT_IN_KERNELS	La liste des noyaux, soutenus par le dispositif, divisé par ";" .
CL_DEVICE_OPENCL_C_VERSION	La version maximale soutenue par le compilateur pour ce dispositif. Le format de la version: "OpenCL<space>C<space><major_version.minor_version><space><vendor-specific information> "
CL_ERROR_DESCRIPTION	Description textuelle d'une erreur OpenCL

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
   int cl_ctx;
   string str;
//--- l'initialisation du contexte OpenCL
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL not found");
      return;
     }
//--- déduisons l'information sur la plateforme
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_NAME,str))
      Print("OpenCL platform name: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VENDOR,str))
      Print("OpenCL platform vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VERSION,str))
      Print("OpenCL platform ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_PROFILE,str))
      Print("OpenCL platform profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_EXTENSIONS,str))
      Print("OpenCL platform ext: ",str);
//--- déduisons l'information sur le dispositif
   if(CLGetInfoString(cl_ctx,CL_DEVICE_NAME,str))
      Print("OpenCL device name: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_PROFILE,str))
      Print("OpenCL device profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_BUILT_IN_KERNELS,str))
      Print("OpenCL device kernels: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_EXTENSIONS,str))
      Print("OpenCL device ext: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VENDOR,str))
      Print("OpenCL device vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VERSION,str))
      Print("OpenCL device ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_OPENCL_C_VERSION,str))
      Print("OpenCL open c ver: ",str);
//--- déduisons l'information totale sur le dispositif  OpenCL
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY));
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE));
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE));
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

© 2000-2024, MetaQuotes Ltd.

CLContextCreate

Crée le contexte OpenCL et retourne un handler sur ce contexte.

```
int  CLContextCreate(
   int  device=CL_USE_ANY     // le numéro de série du dispositif OpenCL ou de la maci
   );
```

Paramètres

device

[in] Le nombre ordinal du périphérique OpenCL dans le système. Au lieu d'un numéro spécifique, vous pouvez spécifier l'une des valeurs suivantes :

- CL_USE_ANY – tout dispositif disponible avec un support OpenCL est autorisé ;
- CL_USE_CPU_ONLY – seule l'émulation OpenCL sur CPU est autorisée ;
- CL_USE_GPU_ONLY – l'émulation OpenCL est interdite et seuls les appareils spécialisés avec support OpenCL (cartes vidéo) peuvent être utilisés;
- CL_USE_GPU_DOUBLE_ONLY – seules les GPUs avec le support du type double sont autorisées.

Valeur de retour

Le handler sur le contexte OpenCL en cas de succès, -1 en cas d'erreur. Pour obtenir des informations sur l'erreur, utilisez la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLContextFree

Supprime le contexte OpenCL.

```
void  CLContextFree(
   int  context     // le handle sur le cotexte OpenCL
   );
```

Paramètres

context

[in] Le handle du cotexte OpenCL.

La valeur rendue

Non. En cas de l'erreur intérieure change la valeur _LastError. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLGetDeviceInfo

Reçoit la propriété du dispositif du pilote informatique OpenCL.

```
bool  CLGetDeviceInfo(
   int     handle,          // le handle du dispositif OpenCL
   int     property_id,     // l'identificateur de la propriété demandée
   uchar&  data[],          // le tableau pour la réception des données
   uint&   size             // le décalage dans le tableau dans les éléments, par défa
   );
```

Paramètres

handle

[in] Le numéro du dispositif OpenCL ou le handle OpenCL, créé par la fonctionCLContextCreate().

property_id

[in] L'identificateur de la propriété, dont il est nécessaire de recevoir sur le dispositif OpenCL. Peut être une des valeurs prédéterminées énumérées au tableau plus bas.

data[]

[out] Le tableau pour la réception des données sur la propriété demandée.

size

[out] La taille des données reçues dans le tableau*data[]*.

La valeur rendue

En cas de l'exécution successive rend true, autrement rend false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour les tableaux unidimentionnels le numéro de l'élément, par qui commence la lecture des données pour l'inscription au tampon OpenCL, est calculé en prenant en compte le drapeau AS_SERIES.

La liste des identificateurs admissibles des propriétés OpenCL du dispositif

On peut trouver la description exacte de la propriété et sa destination sur le site officiel OpenCL.

Identificateur	La valeur
CL_DEVICE_TYPE	0x1000
CL_DEVICE_VENDOR_ID	0x1001
CL_DEVICE_MAX_COMPUTE_UNITS	0x1002
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS	0x1003
CL_DEVICE_MAX_WORK_GROUP_SIZE	0x1004
CL_DEVICE_MAX_WORK_ITEM_SIZES	0x1005
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR	0x1006

© 2000-2024, MetaQuotes Ltd.

Identificateur	La valeur
CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT	0x1007
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT	0x1008
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG	0x1009
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT	0x100A
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE	0x100B
CL_DEVICE_MAX_CLOCK_FREQUENCY	0x100C
CL_DEVICE_ADDRESS_BITS	0x100D
CL_DEVICE_MAX_READ_IMAGE_ARGS	0x100E
CL_DEVICE_MAX_WRITE_IMAGE_ARGS	0x100F
CL_DEVICE_MAX_MEM_ALLOC_SIZE	0x1010
CL_DEVICE_IMAGE2D_MAX_WIDTH	0x1011
CL_DEVICE_IMAGE2D_MAX_HEIGHT	0x1012
CL_DEVICE_IMAGE3D_MAX_WIDTH	0x1013
CL_DEVICE_IMAGE3D_MAX_HEIGHT	0x1014
CL_DEVICE_IMAGE3D_MAX_DEPTH	0x1015
CL_DEVICE_IMAGE_SUPPORT	0x1016
CL_DEVICE_MAX_PARAMETER_SIZE	0x1017
CL_DEVICE_MAX_SAMPLERS	0x1018
CL_DEVICE_MEM_BASE_ADDR_ALIGN	0x1019
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE	0x101A
CL_DEVICE_SINGLE_FP_CONFIG	0x101B
CL_DEVICE_GLOBAL_MEM_CACHE_TYPE	0x101C
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE	0x101D
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE	0x101E
CL_DEVICE_GLOBAL_MEM_SIZE	0x101F
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE	0x1020
CL_DEVICE_MAX_CONSTANT_ARGS	0x1021
CL_DEVICE_LOCAL_MEM_TYPE	0x1022

© 2000-2024, MetaQuotes Ltd.

Identificateur	La valeur
CL_DEVICE_LOCAL_MEM_SIZE	0x1023
CL_DEVICE_ERROR_CORRECTION_SUPPORT	0x1024
CL_DEVICE_PROFILING_TIMER_RESOLUTION	0x1025
CL_DEVICE_ENDIAN_LITTLE	0x1026
CL_DEVICE_AVAILABLE	0x1027
CL_DEVICE_COMPILER_AVAILABLE	0x1028
CL_DEVICE_EXECUTION_CAPABILITIES	0x1029
CL_DEVICE_QUEUE_PROPERTIES	0x102A
CL_DEVICE_NAME	0x102B
CL_DEVICE_VENDOR	0x102C
CL_DRIVER_VERSION	0x102D
CL_DEVICE_PROFILE	0x102E
CL_DEVICE_VERSION	0x102F
CL_DEVICE_EXTENSIONS	0x1030
CL_DEVICE_PLATFORM	0x1031
CL_DEVICE_DOUBLE_FP_CONFIG	0x1032
CL_DEVICE_PREFERRED_VECTOR_WIDTH_HALF	0x1034
CL_DEVICE_HOST_UNIFIED_MEMORY	0x1035
CL_DEVICE_NATIVE_VECTOR_WIDTH_CHAR	0x1036
CL_DEVICE_NATIVE_VECTOR_WIDTH_SHORT	0x1037
CL_DEVICE_NATIVE_VECTOR_WIDTH_INT	0x1038
CL_DEVICE_NATIVE_VECTOR_WIDTH_LONG	0x1039
CL_DEVICE_NATIVE_VECTOR_WIDTH_FLOAT	0x103A
CL_DEVICE_NATIVE_VECTOR_WIDTH_DOUBLE	0x103B
CL_DEVICE_NATIVE_VECTOR_WIDTH_HALF	0x103C
CL_DEVICE_OPENCL_C_VERSION	0x103D
CL_DEVICE_LINKER_AVAILABLE	0x103E
CL_DEVICE_BUILT_IN_KERNELS	0x103F
CL_DEVICE_IMAGE_MAX_BUFFER_SIZE	0x1040
CL_DEVICE_IMAGE_MAX_ARRAY_SIZE	0x1041

© 2000-2024, MetaQuotes Ltd.

Identificateur	La valeur
CL_DEVICE_PARENT_DEVICE	0x1042
CL_DEVICE_PARTITION_MAX_SUB_DEVICES	0x1043
CL_DEVICE_PARTITION_PROPERTIES	0x1044
CL_DEVICE_PARTITION_AFFINITY_DOMAIN	0x1045
CL_DEVICE_PARTITION_TYPE	0x1046
CL_DEVICE_REFERENCE_COUNT	0x1047
CL_DEVICE_PREFERRED_INTEROP_USER_SYNC	0x1048
CL_DEVICE_PRINTF_BUFFER_SIZE	0x1049
CL_DEVICE_IMAGE_PITCH_ALIGNMENT	0x104A
CL_DEVICE_IMAGE_BASE_ADDRESS_ALIGNMENT	0x104B

Exemple:

```
void OnStart()
  {
//---
   int dCount= CLGetInfoInteger(0,CL_DEVICE_COUNT);
   for(int i = 0; i<dCount; i++)
     {
      int clCtx=CLContextCreate(i);
      if(clCtx == -1)
         Print("ERROR in CLContextCreate");
      string device;
      CLGetInfoString(clCtx,CL_DEVICE_NAME,device);
      Print(i,": ",device);
      uchar data[1024];
      uint size;
      CLGetDeviceInfo(clCtx,CL_DEVICE_VENDOR,data,size);
      Print("size = ",size);
      string str=CharArrayToString(data);
      Print(str);
     }
  }
//--- l'exemple de la sortie au journal "Experts"
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      2: Advanced Micro Devices, Inc.
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 32
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Tahiti
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Intel(R) Corporation
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 21
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      1:        Intel(R) Core(TM) i7-37
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      NVIDIA Corporation
```

© 2000-2024, MetaQuotes Ltd.

```
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 19
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      0: GeForce GTX 580
```

© 2000-2024, MetaQuotes Ltd.

CLProgramCreate

Crée un programme OpenCL à partir du code source.

```
int  CLProgramCreate(
   int           context,     // le handle sur le contexte OpenCL
   const string  source       // le code source
   );
```

Paramètres

context

[in] Le handle du contexte OpenCL.

source

[in] La chaîne avec le code source du programme OpenCL.

&build_log

[in] Une chaîne pour recevoir les messages du compilateur OpenCL.

Valeur de retour

Le handle sur le buffer OpenCL en cas de succès. En cas d'erreur, -1 est retourné. Pour obtenir des informations sur l'erreur, utilisez la fonction GetLastError().

Note

À l'heure actuelle, les codes d'erreur suivants sont utilisés :

- ERR_OPENCL_INVALID_HANDLE - handle non valide sur le contexte OpenCL,
- ERR_INVALID_PARAMETER - paramètre de chaîne non valide,
- ERR_NOT_ENOUGH_MEMORY - pas assez de mémoire pour terminer l'opération,
- ERR_OPENCL_PROGRAM_CREATE - erreur interne de l'OpenCL ou erreur de compilation.

Dans certaines cartes graphiques travailler avec les numéros de type double est désactivé par défaut. Cela peut conduire à des erreurs de compilation 5105. Pour activer le support pour les numéros de type double, veuillez ajouter la directive suivante à votre programme OpenCL : #pragma OpenCL EXTENSION cl_khr_fp64 : enable Toutefois, si une carte graphique ne supporte pas les doubles, l'activation de cette directive ne sera pas d'une grande aide.

Exemple :

```
//+------------------------------------------------------------------+
//| Noyau OpenCL                                                     |
//+------------------------------------------------------------------+
const string
cl_src=
       //--- par défaut certains GPU ne supportent pas les doubles
       //--- la directive cl_khr_fp64 est utilisée pour permettre un travail avec des
       "#pragma OPENCL EXTENSION cl_khr_fp64 : enable      \r\n"
       //--- fonction du noyau OpenCL
       "__kernel void Test_GPU(__global double *data,      \r\n"
       "                       const    int N,             \r\n"
       "                       const    int total_arrays)  \r\n"
```

© 2000-2024, MetaQuotes Ltd.

```
        "  {                                                    \r\n"
        "   uint kernel_index=get_global_id(0);                 \r\n"
        "   if (kernel_index>total_arrays) return;              \r\n"
        "   uint local_start_offset=kernel_index*N;             \r\n"
        "   for(int i=0; i<N; i++)                              \r\n"
        "     {                                                 \r\n"
        "       data[i+local_start_offset] *= 2.0;              \r\n"
        "     }                                                 \r\n"
        "  }                                                    \r\n";
//+------------------------------------------------------------------+
//| Test_CPU                                                         |
//+------------------------------------------------------------------+
bool Test_CPU(double &data[],const int N,const int id,const int total_arrays)
  {
//--- vérifie la taille du tableau
   if(ArraySize(data)==0) return(false);
//--- vérifie l'index du tableau
   if(id>total_arrays) return(false);
//--- calcule le décalage pour un tableau avec l'index id
   int local_start_offset=id*N;
//--- multiplie les éléments par 2
   for(int i=0; i<N; i++)
     {
      data[i+local_start_offset]*=2.0;
     }
   retourne vrai;
  }
//---
#define ARRAY_SIZE   100  // taille du tableau
#define TOTAL_ARRAYS 5    // tableaux totaux
//--- Handles OpenCL
int cl_ctx;  // Handle du contexte OpenCL
int cl_prg;  // Handle du programme OpenCL
int cl_krn;  // Handle du noyay OpenCL
int cl_mem;  // Handle du buffer OpenCL
//---
double DataArray1[]; // tableau de données pour le calcul du CPU
double DataArray2[]; // tableau de données pour le calcul du GPU
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- initialisation des objets OpenCL
//--- création du contexte OpenCL
   if((cl_ctx=CLContextCreate())==INVALID_HANDLE)
     {
      Print("OpenCL not found. Erreur=",GetLastError());
      return(1);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- création du programme OpenCL
   if((cl_prg=CLProgramCreate(cl_ctx,cl_src))==INVALID_HANDLE)
      {
       CLContextFree(cl_ctx);
       Print("Le programme OpenCL crée a échoué. Erreur=",GetLastError());
       return(1);
      }
//--- créer le noyau OpenCL
   if((cl_krn=CLKernelCreate(cl_prg,"Test_GPU"))==INVALID_HANDLE)
      {
       CLProgramFree(cl_prg);
       CLContextFree(cl_ctx);
       Print("Le noyau OpenCL crée a échoué. Erreur=",GetLastError());
       return(1);
      }
//--- créer un OpenCL buffer
   if((cl_mem=CLBufferCreate(cl_ctx,ARRAY_SIZE*TOTAL_ARRAYS*sizeof(double),CL_MEM_REAI
      {
       CLKernelFree(cl_krn);
       CLProgramFree(cl_prg);
       CLContextFree(cl_ctx);
       Print("L'OpenCL buffer crée a échoué. Erreur=",GetLastError());
       return(1);
      }
//--- définir les paramêtres du noyau OpenCL
   CLSetKernelArgMem(cl_krn,0,cl_mem);
   CLSetKernelArg(cl_krn,1,ARRAY_SIZE);
   CLSetKernelArg(cl_krn,2,TOTAL_ARRAYS);
//--- préparer des tableaux de données
   ArrayResize(DataArray1,ARRAY_SIZE*TOTAL_ARRAYS);
   ArrayResize(DataArray2,ARRAY_SIZE*TOTAL_ARRAYS);
//--- compléter des tableaux de données
   for(int j=0; j<TOTAL_ARRAYS; j++)
      {
       //--- calcule le décalage pour un tableau journalier
       uint local_offset=j*ARRAY_SIZE;
       //--- préparer le tableau avec index j
       for(int i=0; i<ARRAY_SIZE; i++)
         {
          //--- compléter les tableaux avec la fonction MathCos(i+j);
          DataArray1[i+local_offset]=MathCos(i+j);
          DataArray2[i+local_offset]=MathCos(i+j);
         }
      };
//--- calcul du test CPU
   for(int j=0; j<TOTAL_ARRAYS; j++)
      {
       //--- calcul du tableau avec l'indice j
```

© 2000-2024, MetaQuotes Ltd.

```
      Test_CPU(DataArray1,ARRAY_SIZE,j,TOTAL_ARRAYS);
     }
//--- préparer les paramètres CLExecute
   uint  offset[]={0};
//--- taille globale du travail
   uint  work[]={TOTAL_ARRAYS};
//--- écrit des données sur l'OpenCL buffer
   CLBufferWrite(cl_mem,DataArray2);
//---exécuter le noyau OpenCL
   CLExecute(cl_krn,1,offset,work);
//--- lit des données sur l'OpenCL buffer
   CLBufferRead(cl_mem,DataArray2);
//--- erreur totale
   double total_error=0;
//--- comparer les résultats et calculer l'erreur
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calcule le décalage pour un tableau journalier
      uint local_offset=j*ARRAY_SIZE;
      //--- comparer les résultats
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         double v1=DataArray1[i+local_offset];
         double v2=DataArray2[i+local_offset];
         double delta=MathAbs(v2-v1);
         total_error+=delta;
         //--- montrer les premiers et les derniers tableaux
         if((j==0) || (j==TOTAL_ARRAYS-1))
            PrintFormat("array %d of %d, element [%d]:  %f, %f, [error]=%f",j+1,TOTAL_
        }
     }
   PrintFormat("Total error: %f",total_error);
//--- supprimer des objets OpenCL
//--- OpenCL buffer gratuit
   CLBufferFree(cl_mem);
//--- sans noyau OpenCL
   CLKernelFree(cl_krn);
//--- sans programme OpenCL
   CLProgramFree(cl_prg);
//--- sans contexte OpenCL
   CLContextFree(cl_ctx);
//---
   retourne(0);
  }
```

© 2000-2024, MetaQuotes Ltd.

CLProgramFree

Supprime le programme OpenCL.

```
void  CLProgramFree(
   int  program     // le handle sur l'objet OpenCL
   );
```

Paramètres

program

[in] Le handle de l'objet OpenCL.

La valeur rendue

Non. En cas de l'erreur intérieure change la valeur _LastError. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLKernelCreate

Crée le point de l'entrée au programme OpenCL et rend le handle sur lui.

```
int  CLKernelCreate(
   int           program,        // le handle sur l'objet OpenCL
   const string  kernel_name     // le nom de la fonction est "kernel"
   );
```

Paramètres

program

[in] Le handle sur l'objet du programme OpenCL.

kernel_name

[in] Le nom de la fonction est "kernel", c'est-à-dire le nom du point de l'entrée au programme OpenCL correspondant, par qui commence l'exécution.

La valeur rendue

Le handle sur le tampon OpenCL à l'exécution réussie. Dans le cas de l'erreur rend -1. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour ce moment on prévoit les codes suivants des erreurs:

- ERR_OPENCL_INVALID_HANDLE - le handle non valide sur *program* OpenCL,
- ERR_INVALID_PARAMETER - le paramètre de chaîne non valide,
- ERR_OPENCL_TOO_LONG_KERNEL_NAME - le nom du kernel, contient plus de 127 caractères,
- ERR_OPENCL_KERNEL_CREATE - l'erreur intérieure à la création de l'objet OpenCL.

© 2000-2024, MetaQuotes Ltd.

CLKernelFree

Supprime la fonction du lancement d' OpenCL.

```
void  CLKernelFree(
   int  kernel      // le handle sur le kernel du programme OpenCL
   );
```

Paramètres

kernel_name

[in] Le handle de l'objet du kernel.

La valeur rendue

Non. En cas de l'erreur intérieure change la valeur _LastError. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLSetKernelArg

Expose le paramètre de la fonction OpenCL.

```
bool  CLSetKernelArg(
   int   kernel,       // le handle sur le kernel du programme OpenCL
   uint  arg_index,    // le numéro de l'argument de la fonction OpenCL
   void  arg_value     // le code initial
   );
```

Paramètres

kernel

[in] Le handle sur le kernel du programme OpenCL.

arg_index

[in] Le numéro de l'argument de la fonction, le numérotage commence par zéro.

arg_value

[in] La valeur de l'argument de la fonction.

La valeur rendue

En cas de l'exécution successive rend true, autrement rend false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour ce moment on prévoit les codes suivants des erreurs:

- ERR_INVALID_PARAMETER,
- ERR_OPENCL_INVALID_HANDLE – le handle non valide sur le kernel OpenCL,
- ERR_OPENCL_SET_KERNEL_PARAMETER - l'erreur intérieure OpenCL.

© 2000-2024, MetaQuotes Ltd.

CLSetKernelArgMem

Expose le tampon OpenCL comme un paramètre de la fonction OpenCL.

```
bool  CLSetKernelArgMem(
   int   kernel,            // le handle sur le kernel du programme OpenCL
   uint  arg_index,         // le numéro de l'argument de la fonction OpenCL
   int   cl_mem_handle,     // le handle sur le tampon OpenCL
   );
```

Paramètres

kernel

[in] Le handle sur le kernel du programme OpenCL.

arg_index

[in] Le numéro de l'argument de la fonction, le numérotage commence par zéro.

cl_mem_handle

[in] Le handle sur le tampon OpenCL.

La valeur rendue

En cas de l'exécution successive rend true, autrement rend false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLSetKernelArgMemLocal

Définit le buffer local comme un argument de la fonction du noyau.

```
bool  CLSetKernelArgMemLocal(
   int    kernel,           // handle sur le noyau d'un programme OpenCL
   uint   arg_index,        // indice de l'argument de la fonction OpenCL
   ulong  local_mem_size    // taille du buffer
   );
```

Paramètres

kernel

[in] Handle sur le noyau d'un programme OpenCL.

arg_index

[in] L'indice de l'argument de la fonction, la numérotation commence à zéro.

local_mem_size

[in] Taille du buffer en octets.

Valeur de Retour

Retourne true en cas de succès, sinon retourne false. Pour plus d'informations sur l'erreur, utiliser la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLBufferCreate

Crée le tampon OpenCL et rend le handle sur lui.

```
int  CLBufferCreate(
   int   context,     // le handle sur le contexte OpenCL
   uint  size         // la taille du tampon
   uint  flags        // les propriétés du tampon spécifiées par la combinaison des d
   );
```

Paramètres

context

[in] Le handle sur le contexte OpenCL.

size

[in] La taille du tampon en bytes.

flags

[in] Les propriétés du tampon spécifiées par la combinaison des drapeaux: CL_MEM_READ_WRITE, CL_MEM_WRITE_ONLY, CL_MEM_READ_ONLY, CL_MEM_ALLOC_HOST_PTR.

La valeur rendue

Le handle sur le tampon OpenCL à l'exécution réussie. Dans le cas de l'erreur rend -1. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour ce moment on prévoit les codes suivants des erreurs:

- ERR_OPENCL_INVALID_HANDLE - le handle non valide sur le contexte OpenCL,
- ERR_NOT_ENOUGH_MEMORY - pas assez de mémoire,
- ERR_OPENCL_BUFFER_CREATE - l'erreur intérieure de la création du tampon.

© 2000-2024, MetaQuotes Ltd.

CLBufferFree

Supprime le tampon OpenCL.

```
void  CLBufferFree(
   int   buffer     // le handle sur le tampon OpenCL
   );
```

Paramètres

buffer

[in] Le handle sur le tampon OpenCL.

La valeur rendue

Non. En cas de l'erreur intérieure change la valeur _LastError. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLBufferWrite

Inscrit le tableau au tampon OpenCL et rend la quantité d'éléments inscrits.

```
uint  CLBufferWrite(
   int          buffer,                    // le handle sur le tampon OpenCL
   const void&  data[],                    // le tableau des valeurs
   uint         buffer_offset=0,           // le décalage dans le tampon OpenCL en oct
   uint         data_offset=0,             // le décalage dans le tampon dans les élér
   uint         data_count=WHOLE_ARRAY     // le nombre des valeurs du tableau pour l'
   );
```

Il existe d'autres versions pour l'utilisation de matrices et vecteurs.

Ecrit les valeurs de la matrice dans le buffer et retourne true en cas de succès.

```
uint  CLBufferWrite(
   int           buffer,                    // handle vers le buffer OpenCL
   uint          buffer_offset,             // décalage dans le buffer OpenCL en octet
   matrix<T>     &mat                       //la matrice des valeurs à écrire dans le
   );
```

Ecrit les valeurs du vecteur dans le buffer et retourne true en cas de succès.

```
uint  CLBufferWrite(
   int           buffer,                    // handle vers le buffer OpenCL
   uint          buffer_offset,             // décalage dans le buffer OpenCL en octet
   vector<T>     &vec                       // le vecteur de valeurs à écrire dans le
   );
```

Paramètres

buffer

[in] Le handle du tampon OpenCL.

data[]

[in] le tableau des valeurs, lesquelles il est nécessaire d'inscrire au tampon OpenCL. est transmis selon la référence.

buffer_offset

[in] le décalage dans le tampon OpenCL en octets, par lequel commence l'inscription. Par défaut l'inscription va dès le début du tampon.

data_offset

[in] L'index du premier élément du tableau, à partir de lequel les valeurs sont tirées du tableau pour l'inscription au tampon OpenCL. Par défaut on prend les valeurs dès le début du tableau.

data_count

[in] Le nombre de valeurs, lesquelles il faut inscrire. Toutes les valeurs du tableau par défaut.

mat

© 2000-2024, MetaQuotes Ltd.

[out] La matrice pour les lire les données du buffer peut être de 3 types : matrix, matrixf ou matrixc.

vec

[out] Le vecteur pour les lire les données du buffer peut être de 3 types : vector, vectorf ou vectorc.

La valeur rendue

Le nombre d'éléments inscrits, en cas de l'erreur 0 se revient. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

true si une matrice ou un vecteur est géré avec succès, sinon false.

Note

Pour les tableaux unidimentionnels le numéro de l'élément, par qui commence la lecture des données pour l'inscription au tampon OpenCL, est calculé en prenant en compte le drapeau AS_SERIES.

Le tableau avec la dimension deux et plus est présenté comme unidimentionnel. Dans ce cas *data_offset* - c'est une quantité d'éléments, qu'il faut manquer dans la représentation, et non le nombre d'éléments dans une première dimension.

Exemple de multiplication d'une matrice avec la méthode MatMul et le calcul en parallèle avec OpenCL

```
#define M       3000      // le nombre de lignes de la première matrice
#define K       2000      // le nombre de colonnes de la première matrice est égal au
#define N       3000      // le nombre de colonnes de la deuxième matrice

//+------------------------------------------------------------------+
const string clSrc=
  "#define N     "+IntegerToString(N)+"                              \r\n"
  "#define K     "+IntegerToString(K)+"                              \r\n"
  "                                                                 \r\n"
  "__kernel void matricesMul( __global float *in1,                  \r\n"
  "                           __global float *in2,                  \r\n"
  "                           __global float *out  )                \r\n"
  "{                                                                \r\n"
  "  int m = get_global_id( 0 );                                    \r\n"
  "  int n = get_global_id( 1 );                                    \r\n"
  "  float sum = 0.0;                                               \r\n"
  "  for( int k = 0; k < K; k ++ )                                  \r\n"
  "     sum += in1[ m * K + k ] * in2[ k * N + n ];                 \r\n"
  "  out[ m * N + n ] = sum;                                        \r\n"
  "}                                                                \r\n";
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- initialise le générateur de nombres aléatoires
```

© 2000-2024, MetaQuotes Ltd.

```
  MathSrand((int)TimeCurrent());
//--- remplit les matrices de la taille donnée avec des valeurs aléatoires
  matrixf mat1(M, K, MatrixRandom) ;    // première matrice
  matrixf mat2(K, N, MatrixRandom);     // deuxième matrice

//--- calcule le produit des matrices avec une méthode naïve
  uint start=GetTickCount();
  matrixf matrix_naive=matrixf::Zeros(M, N);// le résultat de la multiplication des 2
  for(int m=0; m<M; m++)
    for(int k=0; k<K; k++)
      for(int n=0; n<N; n++)
        matrix_naive[m][n]+=mat1[m][k]*mat2[k][n];
  uint time_naive=GetTickCount()-start;

//--- calcule le produit des matrices avec MatMull
  start=GetTickCount();
  matrixf matrix_matmul=mat1.MatMul(mat2);
  uint time_matmul=GetTickCount()-start;

//--- calcule le produit des matrices avec OpenCL
  matrixf matrix_opencl=matrixf::Zeros(M, N);
  int cl_ctx;             // handle du contexte
  if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL non trouvé, fin");
    return;
   }
  int cl_prg;             // handle du programme
  int cl_krn;             // handle du noyau
  int cl_mem_in1;         // handle du premier buffer (entrées)
  int cl_mem_in2;         // handle du deuxième buffer (entrées)
  int cl_mem_out;         // handle du troisième buffer (sorties)
//--- crée le programme et le noyau
  cl_prg = CLProgramCreate(cl_ctx, clSrc);
  cl_krn = CLKernelCreate(cl_prg, "matricesMul");
//--- crée les 3 buffers pour les 3 matrices
  cl_mem_in1=CLBufferCreate(cl_ctx, M*K*sizeof(float), CL_MEM_READ_WRITE);
  cl_mem_in2=CLBufferCreate(cl_ctx, K*N*sizeof(float), CL_MEM_READ_WRITE);
//--- 3ème matrice - sorties
  cl_mem_out=CLBufferCreate(cl_ctx, M*N*sizeof(float), CL_MEM_READ_WRITE);
//--- définit les arguments du noyau
  CLSetKernelArgMem(cl_krn, 0, cl_mem_in1);
  CLSetKernelArgMem(cl_krn, 1, cl_mem_in2);
  CLSetKernelArgMem(cl_krn, 2, cl_mem_out);
//--- écrit les matrices dans le buffer du périphérique
  CLBufferWrite(cl_mem_in1, 0, mat1);
  CLBufferWrite(cl_mem_in2, 0, mat2);
  CLBufferWrite(cl_mem_out, 0, matrix_opencl);
//--- heure de démarrage de l'exécution du code OpenCL
```

© 2000-2024, MetaQuotes Ltd.

```
  start=GetTickCount();
//--- définit les paramètres de la zone de travail de la tâche et exécute le programme
  uint  offs[2] = {0, 0};
  uint works[2] = {M, N};
  start=GetTickCount();
  bool ex=CLExecute(cl_krn, 2, offs, works);
//--- calcule le résultat dans la matrice
  if(CLBufferRead(cl_mem_out, 0, matrix_opencl))
    PrintFormat("[%d x %d] matrice lue : ", matrix_opencl.Rows(), matrix_opencl.Cols()
   else
      Print("CLBufferRead(cl_mem_out, 0, matrix_opencl a échoué. Erreur ",GetLastError
  uint time_opencl=GetTickCount()-start;
  Print("Comparaison des temps de calcul de chaque méthode");
  PrintFormat("Produit naïf = %d ms",time_naive);
  PrintFormat("Produit MatMul = %d ms",time_matmul);
  PrintFormat("Produit OpenCl = %d ms",time_opencl);
//--- libère tous les contextes OpenCL
  CLFreeAll(cl_ctx, cl_prg, cl_krn, cl_mem_in1, cl_mem_in2, cl_mem_out);

//--- compare les matrices de résultats obtenues
  Print("Combien y a-t-il d'erreurs de divergence entre les matrices de résultats ?");
  ulong errors=matrix_naive.Compare(matrix_matmul,(float)1e-12);
  Print("matrix_direct.Compare(matrix_matmul,1e-12)=",errors);
  errors=matrix_matmul.Compare(matrix_opencl,float(1e-12));
  Print("matrix_matmul.Compare(matrix_opencl,1e-12)=",errors);
/*
  Résultat :

   [3000 x 3000] matrice lue :
  Comparaison des temps de calcul de chaque méthode
   Produit naïf = 54750 ms
   Produit MatMul  = 4578 ms
   Produit OpenCl  = 922 ms
  Combien y a-t-il d'erreurs de divergence entre les matrices de résultats ?
   matrix_direct.Compare(matrix_matmul,1e-12)=0
   matrix_matmul.Compare(matrix_opencl,1e-12)=0
*/
 }
//+------------------------------------------------------------------+
//| Remplit la matrice avec des valeurs aléatoires                   |
//+------------------------------------------------------------------+
void MatrixRandom(matrixf& m)
 {
  for(ulong r=0; r<m.Rows(); r++)
   {
    for(ulong c=0; c<m.Cols(); c++)
     {
      m[r][c]=(float)((MathRand()-16383.5)/32767.);
     }
```

```
    }
  }
//+------------------------------------------------------------------+
//| Libère tous les contextes OpenCL                                 |
//+------------------------------------------------------------------+
void CLFreeAll(int cl_ctx, int cl_prg, int cl_krn,
               int cl_mem_in1, int cl_mem_in2, int cl_mem_out)
  {
//--- efface tous les contextes créés par OpenCL en ordre inverse
   CLBufferFree(cl_mem_in1);
   CLBufferFree(cl_mem_in2);
   CLBufferFree(cl_mem_out);
   CLKernelFree(cl_krn);
   CLProgramFree(cl_prg);
   CLContextFree(cl_ctx);
  }
```

© 2000-2024, MetaQuotes Ltd.

CLBufferRead

Lit le tampon OpenCL au tableau et rend la quantité d'éléments lus.

```
uint  CLBufferRead(
   int          buffer,                    // le handle sur le tampon OpenCL
   const void&  data[],                    // le tableau des valeurs
   uint         buffer_offset=0,           // le décalage dans le tampon OpenCL en oct
   uint         data_offset=0,             // le décalage dans le tampon dans les élér
   uint         data_count=WHOLE_ARRAY     // le nombre des valeurs du tableau pour la
   );
```

Il existe d'autres versions pour l'utilisation de matrices et vecteurs.

Lit le buffer OpenCL et l'écrit dans la matrice et retourne true en cas de succès.

```
uint  CLBufferRead(
   int           buffer,                   // handle vers le buffer OpenCL
   uint          buffer_offset,            // décalage dans le buffer OpenCL en octet
   const matrxi& mat,                      // la matrice dans laquelle les valeurs du
   ulong         rows=-1,                  // le nombre de lignes de la matrice
   ulong         cols=-1                   // le nombre de colonnes de la matrice
   );
```

Lit le buffer OpenCL et l'écrit dans le vecteur et retourne true en cas de succès.

```
uint  CLBufferRead(
   int           buffer,                   // handle vers le buffer OpenCL
   uint          buffer_offset,            // décalage dans le buffer OpenCL en octet
   const vector& vec,                      // le vecteur dans lequel les valeurs du b
   ulong         size-1,                   // longueur du vecteur
   );
```

Paramètres

buffer

[in] Le handle du tampon OpenCL.

data[]

[in] Le massif pour la réception des valeurs du tampon OpenCL. est transmis selon la référence.

buffer_offset

[in] Le décalage dans le tampon OpenCL en octets, par lequel commence la lecture. Par défaut la lecture commence par le début du tampon.

data_offset

[in] L'index du premier élément du massif pour l'inscription des valeurs du tampon OpenCL. Par défaut la lecture l'inscription des valeurs lues au tableau commence par index nul.

data_count

[in] Le nombre de valeurs, lesquelles il faut lire. Tout le tampon OpenCL est lu par défaut.

mat

© 2000-2024, MetaQuotes Ltd.

[out] La matrice pour les lire les données du buffer peut être de 3 types : matrix, matrixf ou matrixc.

vec

[out] Le vecteur pour les lire les données du buffer peut être de 3 types : vector, vectorf ou vectorc.

rows=-1

[in] Si le paramètre est spécifié, le paramètre *cols* doit également être spécifié. Si les dimensions de la nouvelle matrice ne sont pas spécifiées, les dimensions courantes seront utilisées. Si la valeur est -1, alors le nombre de lignes ne changera pas.

cols=-1

[in] Si le paramètre n'est pas spécifié, le paramètre *rows* doit être spécifié également. La matrice adhère à la règle suivante : soit les 2 paramètres sont spécifiés, soit aucun, ou sinon une erreur est générée. Si les 2 paramètres (*rows et cols*) sont spécifiés, la taille de la matrice sera changée. Si -1, le nombre de colonnes ne sera pas changé.

size=-1

[in] Si le paramètre n'est pas spécifié ou si sa valeur est -1, la longueur du vecteur ne changera pas.

La valeur rendue

Le nombre d'éléments lus, en cas de l'erreur 0 se revient. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

true si une matrice ou un vecteur est géré avec succès, sinon false.

Note

Pour les tableaux unidimentionnels le numéro de l'élément, auquel commence l'inscription des données du tampon OpenCL, est calculé en prenant en compte le drapeau AS_SERIES.

Le tableau avec la dimension deux et plus est présenté comme unidimentionnel. Dans ce cas *data_offset* - c'est une quantité d'éléments, qu'il faut manquer dans la représentation, et non le nombre d'éléments dans une première dimension.

Exemple de calcul de Pi en utilisant l'équation :

$$\pi = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \frac{4}{1+\left(\frac{2k+1}{2N}\right)^2} = 16 \lim_{N\to\infty} N \sum_{k=0}^{N-1} \frac{1}{4N^2+(2k+1)^2}$$

```
#define  _num_steps        1000000000
#define  _divisor          40000
#define  _step             1.0 / _num_steps
#define  _intrnCnt         _num_steps / _divisor
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
string D2S(double arg, int digits) { return DoubleToString(arg, digits); }
string I2S(int arg)                { return IntegerToString(arg); }

//--- code du programme OpenCL
const string clSource=
  "#define _step "+D2S(_step, 12)+"                    \r\n"
  "#define _intrnCnt "+I2S(_intrnCnt)+"                \r\n"
  "                                                    \r\n"
  "__kernel void Pi( __global double *out )            \r\n"
  "{                                                   \r\n"
  "  int i = get_global_id( 0 );                       \r\n"
  "  double partsum = 0.0;                             \r\n"
  "  double x = 0.0;                                   \r\n"
  "  long from = i * _intrnCnt;                        \r\n"
  "  long to = from + _intrnCnt;                       \r\n"
  "  for( long j = from; j < to; j ++ )                \r\n"
  "  {                                                 \r\n"
  "     x = ( j + 0.5 ) * _step;                       \r\n"
  "     partsum += 4.0 / ( 1. + x * x );               \r\n"
  "  }                                                 \r\n"
  "  out[ i ] = partsum;                               \r\n"
  "}                                                   \r\n";

//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
int OnStart()
 {
  Print("Calcul de Pi : pas = "+D2S(_step, 12)+"; _intrnCnt = "+I2S(_intrnCnt));
//--- prépare les contextes OpenCL
  int clCtx;
  if((clCtx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL non trouvé");
    return(-1);
   }
  int clPrg = CLProgramCreate(clCtx, clSource);
  int clKrn = CLKernelCreate(clPrg, "Pi");
  int clMem=CLBufferCreate(clCtx, _divisor*sizeof(double), CL_MEM_READ_WRITE);
  CLSetKernelArgMem(clKrn, 0, clMem);

  const uint offs[1]  = {0};
  const uint works[1] = {_divisor};
//--- lance le programme OpenCL
  ulong start=GetMicrosecondCount();
```

© 2000-2024, MetaQuotes Ltd.

```
  if(!CLExecute(clKrn, 1, offs, works))
   {
    Print("CLExecute(clKrn, 1, offs, works) a échoué ! Erreur ", GetLastError());
    CLFreeAll(clMem, clKrn, clPrg, clCtx);
    return(-1);
   }
//--- récupère les résultats du périphérique OpenCL
  vector buffer(_divisor);
  if(!CLBufferRead(clMem, 0, buffer))
   {
    Print("CLBufferRead(clMem, 0, buffer) a échoué ! Erreur ", GetLastError());
    CLFreeAll(clMem, clKrn, clPrg, clCtx);
    return(-1);
   }
//--- additionne toutes les valeurs pour calculer Pi
  double Pi=buffer.Sum()*_step;

  double time=(GetMicrosecondCount()-start)/1000.;
  Print("OpenCL: Pi calculaté pour "+D2S(, 2)+" ms");
  Print("Pi = "+DoubleToString(Pi, 12));
//--- libère la mémoire
  CLFreeAll(clMem, clKrn, clPrg, clCtx);
//--- succès
  return(0);
 }
  /*
  Calcul de Pi : pas = 0.000000001000; _intrnCnt = 25000
  OpenCL: Périphérique GPU  'Ellesmere' selected
  OpenCL: Pi calculé for 99.98 ms
  Pi = 3.141592653590
  */
//+------------------------------------------------------------------+
//| Routine auxiliaire pour libérer la mémoire                       |
//+------------------------------------------------------------------+
void CLFreeAll(const int clMem, const int clKrn, const int clPrg, const int clCtx)
 {
  CLBufferFree(clMem);
  CLKernelFree(clKrn);
  CLProgramFree(clPrg);
  CLContextFree(clCtx);
 }
```

© 2000-2024, MetaQuotes Ltd.

CLExecute

Exécute le programme OpenCL. Il y a 3 variantes de la fonction:

1. Le lancement de la fonction kernel sur un seul noyau

```
bool  CLExecute(
   int          kernel                  // le handle sur le kernel du programme OpenC
   );
```

2. Le lancement de quelques copies kernel (la fonction OpenCL) avec une description de l'espace des tâches

```
bool  CLExecute(
   int          kernel,                  // le handle sur le kernel du programme Oper
   uint         work_dim,                // la dimension de l'espace des tâches
   const uint&  global_work_offset[],    // le décalage initial dans l'espace des tâc
   const uint&  global_work_size[]       // le nombre total de tâches
   );
```

3. Le lancement de quelques copies kernel (la fonction OpenCL) avec la description de l'espace des tâches et l'indication de la taille de la sous-multitude locale des tâches dans le groupe

```
bool  CLExecute(
   int          kernel,                  // le handle sur le kernel du programme Oper
   uint         work_dim,                // la dimension de l'espace des tâches
   const uint&  global_work_offset[],    // le décalage initial dans l'espace des tâc
   const uint&  global_work_size[],      // le nombre total de tâches
   const uint&  local_work_size[]        // le nombre de tâches dans le groupe local
   );
```

Paramètres

kernel

[in] Le handle sur le kernel OpenCL.

work_dim

[in] La dimension de l'espace des tâches.

global_work_offset[]

[in] Le décalage initial dans l'espace des tâches.

global_work_size[]

[in] La taille d'un sous-ensemble de tâches.

local_work_size[]

[in] La taille de la sous-multitude locale des tâches dans le groupe.

La valeur rendue

En cas de l'exécution successive rend true, autrement rend false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

Note

montrons le sens des paramètres à l'exemple:

- *work_dim* spécifie la dimension du tableau *work_items[]*, décrivant les tâches. Si *work_dim*=3, on utilise le tableau en 3 dimensions *work_items[N1, N2, N3]*.
- *global_work_size[]* contient des valeurs qui spécifient la taille du tableau *work_items[]*. Si work_dim=3, et en conséquence le tableau*global_work_size[3]* peut-être {40, 100, 320}. Alors nous avons *work_items[40, 100, 320]*. Ainsi, le nombre total de tâches est égal à 40 x 100 x 320 = 1 280 000.
- *local_work_size[]* spécifie un sous-ensemble des tâches à exécuter par le noyau OpenCL spécifié du programme. Sa dimension est égale à la dimension *work_items[]* et permet de couper la sous-multitude totale des tâches sur les sous-multitudes plus petites sans restes de la division. En effet, les dimensions du tableau *local_work_size[]* doivent être choisis de sorte que la pluralité de tâches globales *work_items[]* soient coupées en petits sous-ensembles. Dans l'exemple donné convient *local_work_size[3]={10, 10, 10}*, car *work_items[40, 100, 320]*on peut ramasser du tableau*local_items[10, 10, 10]*sans reste.

© 2000-2024, MetaQuotes Ltd.

CLExecutionStatus

Retourne le statut d'exécution du programme OpenCL.

```
int  CLExecutionStatus(
   int   kernel            // handle sur le noyau d'un programme OpenCL
   );
```

Paramètres

kernel

[in] Handle sur le noyau d'un programme OpenCL.

Valeur de Retour

Retourne le statut du programme OpenCL. La valeur peut être l'une des suivantes :

- CL_COMPLETE=0 - programme terminé,
- CL_RUNNING=1 - en cours d'exécution,
- CL_SUBMITTED=2 - soumis pour exécution,
- CL_QUEUED=3 - mis en attente,
- -1 (moins un) - une erreur est survenue lors de l'exécution de CLExecutionStatus().

© 2000-2024, MetaQuotes Ltd.

Utilisation des bases de données

Les fonctions permettant l'utilisation des bases de données sont basées sur le moteur populaire et facile à utiliser SQLite. La fonctionnalité pratique de ce moteur est que la base de données entière se trouve dans un seul fichier sur le disque dur du PC de l'utilisateur.

Les fonctions permettent de créer facilement des tables, d'y ajouter des données, d'effectuer des modifications et d'échantillonner à l'aide de requêtes SQL simples :

- recevoir l'historique des transactions et les cotations de tous les formats,
- sauvegarde des résultats d'optimisation et de test,
- préparer et échanger des données avec d'autres progiciels d'analyse,
- stockage des paramètres et de l'état de l'application MQL5.

Les requêtes permettent d'utiliser les fonctions statistiques et mathématiques.

Les fonctions de travail avec les bases de données vous permettent de remplacer les opérations de gestion des tableaux de données volumineuses les plus répétitives par des requêtes SQL, de sorte qu'il est souvent possible d'utiliser des appels à DatabaseExecute /DatabasePrepare au lieu de programmer des boucles et des comparaisons complexes. Utilisez la fonction DatabaseReadBind pour obtenir facilement les résultats de la requête dans une structure prête à l'emploi. La fonction permet de lire tous les champs d'enregistrement à la fois dans un seul appel.

Pour accélérer la lecture, l'écriture et la modification, une base de données peut être ouverte/créée en RAM avec l'indicateur DATABASE_OPEN_MEMORY, bien qu'une telle base de données ne soit disponible que pour une application spécifique et ne soit pas partagée. Lorsque vous travaillez avec des bases de données situées sur le disque dur, les insertions/modifications de données en masse doivent être encapsulées dans des transactions à l'aide de DatabaseTransactionBegin/DatabaseTransactionCommit/DatabaseTransactionRollback. Cela accélère le processus des centaines de fois.

Pour commencer à travailler avec les fonctions, lisez l'article SQLite: Native handling of SQL databases in MQL5.

Fonction	Action
DatabaseOpen	Ouvre ou crée une base de données dans le fichier spécifié
DatabaseClose	Ferme une base de données
DatabaseImport	Imports data from a file into a table
DatabaseExport	Exports a table or an SQL request execution result to a CSV file
DatabasePrint	Prints a table or an SQL request execution result in the Experts journal
DatabaseTableExists	Vérifie la présence d'une table dans la base de données
DatabaseExecute	Exécute une requête dans la base de données spécifiée
DatabasePrepare	Crée le handle d'une requête, qui peut être ensuite exécutée avec DatabaseRead()

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
DatabaseReset	Réinitialise une requête, comme après avoir appelé DatabasePrepare()
DatabaseBind	Définit la valeur d'un paramètre dans une requête
DatabaseBindArray	Définit un tableau comme valeur de paramètre
DatabaseRead	Se déplace sur l'enregistrement suivant des résultats de la requête
DatabaseReadBind	Moves to the next record and reads data into the structure from it
DatabaseFinalize	Supprime une requête créée avec DatabasePrepare()
DatabaseTransactionBegin	Démarre l'exécution d'une transaction
DatabaseTransactionCommit	Termine l'exécution d'une transaction
DatabaseTransactionRollback	Annule une transaction
DatabaseColumnsCount	Retourne le nombre de colonnes dans une requête
DatabaseColumnName	Retourne le nom d'une colonne par son numéro
DatabaseColumnType	Retourne le type d'une colonne par son numéro
DatabaseColumnSize	Retourne la taille d'une colonne en octets
DatabaseColumnText	Retourne la valeur d'une colonne de type string de l'enregistrement courant
DatabaseColumnInteger	Retourne la valeur d'une colonne de type int de l'enregistrement courant
DatabaseColumnLong	Retourne la valeur d'une colonne de type long de l'enregistrement courant
DatabaseColumnDouble	Retourne la valeur d'une colonne de type double de l'enregistrement courant
DatabaseColumnBlob	Retourne le tableau de caractères d'une colonne de type BLOB de l'enregistrement courant

Fonctions statistiques :

- mode - mode
- médiane - médiane (50e percentile)
- centile_25 - 25e percentile
- percentile_75
- percentile_90
- percentile_95
- percentile_99

© 2000-2024, MetaQuotes Ltd.

- stddev ou stddev_samp - exemple d'écart type
- stddev_pop - écart type de la population
- variance ou var_samp - variance de l'échantillon
- var_pop - variance de la population

Fonctions mathématiques

- acos(X) - arc-cosinus en radians
- acosh(X) - arc-cosinus hyperbolique
- asin(X) - arc-sinus en radians
- asinh(X) - arc-sinus hyperbolique
- atan(X) - arc-tangente en radians
- atan2(X,Y) - arc-tangente en radians du rapport X/Y
- atanh(X) - arc-tangente hyperbolique
- ceil(X) - arrondi à l'entier supérieur
- ceiling(X) - arrondi à l'entier supérieur
- cos(X) - cosinus angulaire en radians
- cosh(X) - cosinus hyperbolique
- degrees(X) - convertir les radians en angle
- exp(X) - exposant
- floor(X) - arrondi à l'entier inférieur
- ln(X) - logarithme naturel
- log(B,X) - logarithme à la base indiquée
- log(X) - logarithme décimal
- log10(X) - logarithme décimal
- log2(X) - logarithme en base 2
- mod(X,Y) - reste de la division
- pi() - Pi approximatif
- pow(X,Y) - puissance par la base indiquée
- power(X,Y) - puissance par la base indiquée
- radians(X) - convertir l'angle en radians
- sin(X) - angle sinusoïdal en radians
- sinh(X) -sinus hyperbolique
- sqrt(X) - racine carrée
- tan(X) - angle tangent en radians
- tanh(X) - tangente hyperbolique
- trunc(X) - tronquer à l'entier le plus proche de 0

Exemple :

```
select
  count(*) as book_count,
```

© 2000-2024, MetaQuotes Ltd.

```
  cast(avg(parent) as integer) as mean,
  cast(median(parent) as integer) as median,
  mode(parent) as mode,
  percentile_90(parent) as p90,
  percentile_95(parent) as p95,
  percentile_99(parent) as p99
from moz_bookmarks;
```

© 2000-2024, MetaQuotes Ltd.

DatabaseOpen

Ouvre ou crée une base de données dans le fichier spécifié.

```
int  DatabaseOpen(
   string  filename,       // nom du fichier
   uint    flags           // combinaison de flags
   );
```

Paramètres

filename

[in] Nom du fichier, relativement au répertoire "MQL5\Files".

flags

[in] Combinaison de flags de l'énumération ENUM_DATABASE_OPEN_FLAGS.

Valeur de Retour

En cas de succès, La fonction retourne le handle sur la base de données, qui peut être ensuite utilisé pour accéder à la base de données. Sinon, elle retourne INVALID_HANDLE. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INTERNAL_ERROR (4001) - erreur critique d'exécution ;
- ERR_WRONG_INTERNAL_PARAMETER (4002) - erreur interne lors de l'accès au répertoire "MQL5\Files" ;
- ERR_INVALID_PARAMETER (4003) - chemin vers le fichier de base de données contenant une chaîne vide, ou la combinaison de flag est invalide ;
- ERR_NOT_ENOUGH_MEMORY (4004) - mémoire insuffisante ;
- ERR_WRONG_FILENAME (5002) - mauvais nom de fichier de base de données ;
- ERR_TOO_LONG_FILENAME (5003) - la taille du chemin absolu vers le fichier de la base de données est supérieure à la longueur auorisée ;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - le nombre d'objets Base de données est supérieur au maximum autorisé ;
- ERR_DATABASE_CONNECT (5123) - erreur de connexion à la base de données;
- ERR_DATABASE_MISUSE (5621) - utilisation incorrecte de la bibliothèque SQLite.

Note

Si le paramètre *filename* comporte NULL ou la chaîne vide "", un fichier temporaire est créé sur le disque. Il est automatiquement supprimé après la fermeture de la connexion à la base de données.

Si le paramètre *filename* comporte ":memory:", la base de données est créée en mémoire et est automatiquement supprimée après la fermeture de la connexion.

Si le paramètre *flags* ne comporte aucun des indicateurs DATABASE_OPEN_READONLY ou DATABASE_OPEN_READWRITE, l'indicateur DATABASE_OPEN_READWRITE est utilisé.

Si l'extension du fichier n'est pas spécifiée, ".sqlite" est utilisé.

ENUM_DATABASE_OPEN_FLAGS

© 2000-2024, MetaQuotes Ltd.

ID	Description
DATABASE_OPEN_READONLY	Lecture seule
DATABASE_OPEN_READWRITE	Ouverture en lecture et en écriture
DATABASE_OPEN_CREATE	Crée le fichier sur le disque si nécessaire
DATABASE_OPEN_MEMORY	Crée la base de données en mémoire
DATABASE_OPEN_COMMON	Le fichier est dans le répertoire commun de tous les terminaux

Voir aussi

DatabaseClose

© 2000-2024, MetaQuotes Ltd.

DatabaseClose

Ferme une base de données.

```
void  DatabaseClose(
   int  database      // handle de la base de données retourné par DatabaseOpen
   );
```

Paramètres

database

[in] Handle de la base de données retourné par DatabaseOpen().

Valeur de Retour

Aucun.

Note

Après l'appel à DatabaseClose, tous les handles de requêtes à la base de données sont automatiquement supprimés et deviennent invalides.

Si le handle est invalide, la fonction retourne l'erreur ERR_DATABASE_INVALID_HANDLE. Vous pouvez vérifier l'erreur en appelant GetLastError().

Voir aussi

DatabaseOpen, DatabasePrepare

© 2000-2024, MetaQuotes Ltd.

DatabaseImport

Importe des données depuis un fichier dans une table.

```
long  DatabaseImport(
   int           database,          // handle de la base de données obtenu avec Databa
   const string  table,             // nom d'une table dans laquelle insérer des donné
   const string  filename,          // nom du fichier contenant les données à importe
   uint          flags,             // combinaison de flags
   const string  separator,         // séparateur de données
   ulong         skip_rows,         // le nombre de lignes à ignorer
   const string  skip_comments      // chaîne de caractères identifiant les commentai
   );
```

Paramètres

database

[in] Handle de la base de données obtenu avec DatabaseOpen().

table

[in] Nom d'une table dans laquelle les données du fichier seront ajoutées.

filename

[in] Fichier CSV ou ZIP contenant les données à lire. Le nom peut contenir des sous-répertoires et est relatif au répertoire MQL5\Files.

flags

[in] Combinaison des flags de l'énumération ENUM_DATABASE_IMPORT_FLAGS.

separator

[in] Séparateur de données dans le fichier CSV.

skip_rows

[in] Nombre de chaînes de caractères à ignorer lors de la lecture des données du fichier.

skip_comments

[in] Chaîne de caractères utilisée pour désigner les commentaires. Si un des caractères de *skip_comments* est détecté au début d'une chaîne, celle-ci sera considérée comme étant un commentaire et ne sera pas importée.

Valeur de Retour

Retourne le nombre de chaînes importées ou -1 en cas d'erreur. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INVALID_PARAMETER (4003) - aucun nom de table spécifiée (chaîne vide ou NULL) ;
- ERR_DATABASE_INTERNAL (5120) - erreur interne de la base de données ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données.

Note

S'il n'y a pas de table nommée *table*, elle est générée automatiquement. Les noms et les types des champs dans la table créée sont définis automatiquement suivant les données du fichier.

© 2000-2024, MetaQuotes Ltd.

ENUM_DATABASE_IMPORT_FLAGS

Identifiant	Description
DATABASE_IMPORT_HEADER	La première ligne contient les noms des champs de la table
DATABASE_IMPORT_CRLF	CRLF (la valeur par défaut est LF) est considéré comme étant un retour à la ligne
DATABASE_IMPORT_APPEND	Ajoute des données à la fin d'une table
DATABASE_IMPORT_QUOTED_STRINGS	Des chaînes de caractères entre guillements (double quotes)
DATABASE_IMPORT_COMMON_FOLDER	Le fichier est stocké dans le répertoire commun à tous les terminaux clients terminals \Terminal\Common\File.

Exemple de lecture d'une table depuis un fichier créé avec du code via l'exemple de DatabaseExport :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
 {
  string csv_filename;
//--- récupèrle noms des fichiers texte à télécharger depuis le répertoire commun des
  string filenames[];
  if(FileSelectDialog("Sélectionner un fichier CSV pour télécharger une table", NULL,
                      "Text files (*.csv)|*.csv",
                      FSD_WRITE_FILE|FSD_COMMON_FOLDER, filenames, "data.csv")>0)
   {
    //--- affiche le nom de chaque fichier sélectionné
    if(ArraySize(filenames)==1)
      csv_filename=filenames[0];
    else
     {
      Print("Erreur inconnue lors de la sélection du fichier. Code d'erreur ", GetLast
      return;
     }
   }
  else
   {
    Print("Fichier CSV non sélectionné");
    return;
   }
//--- crée ou ouvre une base de données
  string db_filename="test.sqlite";
  int db=DatabaseOpen(db_filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- vérifie si la table TEST existe
  if(DatabaseTableExists(db, "TEST"))
   {
    //--- supprime la table TEST
    if(!DatabaseExecute(db, "DROP TABLE IF EXISTS TEST"))
     {
      Print("Echec de la suppression de la table TEST avec le code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   }
  //--- importe les entrées du fichier dans la table TEST
  long imported=DatabaseImport(db, "TEST", csv_filename, DATABASE_IMPORT_HEADER|DATABA
  if(imported>0)
   {
    Print(imported," lignes importées dans la table TEST");
    DatabasePrint(db,"SELECT * FROM TEST",DATABASE_PRINT_NO_INDEX);
   }
  else
    {
     Print("Echec de DatabaseImport(). Erreur ",GetLastError());
    }
//--- ferme la base de données
  DatabaseClose(db);
  PrintFormat("Base de données : %s fermée", db_filename);
 }
```

Voir aussi

DatabaseOpen, DatabasePrint

© 2000-2024, MetaQuotes Ltd.

DatabaseExport

Exporte une table ou le résultat de l'exécution d'une requête SQL dans un fichier CSV. Le fichier est créé avec un encodage en UTF-8.

```
long  DatabaseExport(
   int           database,          // handle de la base de données obtenu avec Datab
   const string  table_or_sql,      // nom d'une table ou requête SQL
   const string  filename,          // nom d'un fichier CSV pour l'export des données
   uint          flags,             // combinaison de flags
   const string  separator          // séparateur de données dans le fichier CSV
   );
```

Paramètres

database

[in] Handle de la base de données obtenu avec DatabaseOpen().

table_or_sql

[in] Le nom d'une table ou le texte d'une requête SQL dont les résultats sont à exporter vers le fichier spécifié.

filename

[in] Nom d'un fichier pour l'export des données. Le chemin est relatif au répertoire MQL5\Files.

flags

[in] Combinaison des flags de l'énumération ENUM_DATABASE_EXPORT_FLAGS.

separator

[in] Séparateur de données. Si NULL est spécifié, le caractère '\t' de tabulation est utilisé comme séparateur. Une chaîne vide "" est considérée comme un séparateur valide, mais le fichier CSV ne pourra pas être lu sous forme de table - il sera considéré comme étant une suite de chaînes de caractères.

Valeur de Retour

Retourne le nombre d'entrées exporteés ou une valeur négative en cas d'erreur. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INTERNAL_ERROR (4001) - erreur critique d'exécution ;
- ERR_INVALID_PARAMETER (4003) - chemin vers le fichier de base de données contenant une chaîne vide, ou combinaison invalide de flags ;
- ERR_NOT_ENOUGH_MEMORY (4004) - mémoire insuffisante ;
- ERR_FUNCTION_NOT_ALLOWED(4014) - le pipe spécifié n'est pas autorisé ;
- ERR_PROGRAM_STOPPED(4022) - opération annulée (le programme MQL s'est stoppé) ;
- ERR_WRONG_FILENAME (5002) - nom de fichier invalide ;
- ERR_TOO_LONG_FILENAME (5003) - la taille du chemin absolu est supérieure à la longueur maximum ;
- ERR_CANNOT_OPEN_FILE(5004) - impossible d'ouvrir le fichier en écriture ;
- ERR_FILE_WRITEERROR(5026) - impossible d'écrire vers le fichier ;

© 2000-2024, MetaQuotes Ltd.

- ERR_DATABASE_INTERNAL (5120) - erreur interne de la base de données ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données ;
- ERR_DATABASE_QUERY_PREPARE(5125) - erreur de génération de la requête ;
- ERR_DATABASE_QUERY_NOT_READONLY - seules les requêtes en lecture seule sont autorisées.

Note

Si les résultats de la requête doivent être exportés, la requête SQL doit commencer avec "SELECT" ou "select". En d'autres termes, la requête SQL ne peut pas modifier le statut de la base de données, sinon DatabaseExport() échouera avec une erreur.

Les valeurs des chaînes de caractères de la base de données peuvent contenir des caractères de conversion ('\r' ou '\r\n'), ainsi que les caractères de séparation définis dans le paramètre *separator*. Dans ce cas, assuez-vous d'utiliser le flag DATABASE_EXPORT_QUOTED_STRINGS dans le paramètre 'flags'. Si ce flag est présent, toutes les chaînes de caractères sont placées entre guillemets. Si une chaîne de caractères contient un guillemet, il sera remplacé par 2 guillemets.

ENUM_DATABASE_EXPORT_FLAGS

Identifiant	Description
DATABASE_EXPORT_HEADER	Affiche les noms des champs dans la première chaîne
DATABASE_EXPORT_INDEX	Affiche les index
DATABASE_EXPORT_NO_BOM	N'insère pas la marque BOM au début du fichier (la BOM est insérée par défaut)
DATABASE_EXPORT_CRLF	Utilise CRLF pour le retour à la ligne (la valeur par défaut sinon est LF)
DATABASE_EXPORT_APPEND	Ajoute les données à la fin d'un fichier existante (par défaut, le fichier est écrasé). Si le fichier n'existe pas, il sera créé.
DATABASE_EXPORT_QUOTED_STRINGS	Exporte les chaînes de caractères entre guillemets.
DATABASE_EXPORT_COMMON_FOLDER	Un fichier CSV est créé dans le dossier commun de tous les terminaux clients \Terminal\Common\File.

Exemple :

```
input int InpRates=100;
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2024, MetaQuotes Ltd.

```
 {
  MqlRates rates[];
//--- stocke l'heure de début avant d'obtenir les barres
  ulong start=GetMicrosecondCount();
//--- récupère les 100 dernières barres sur H1
  if(CopyRates(Symbol(), PERIOD_H1, 1, InpRates, rates)<InpRates)
   {
    Print("Echec de CopyRates(), erreur ", GetLastError());
    return;
   }
  else
   {
    //--- combien de barres ont été récupérées et combien de temps il a fallu pour les
    PrintFormat("%s: CopyRates a reçu %d barres en %d ms ",
                _Symbol, ArraySize(rates), (GetMicrosecondCount()-start)/1000);
   }
//--- définit le nom du fichier pour stocker la base de données
  string filename=_Symbol+"_"+EnumToString(PERIOD_H1)+"_"+TimeToString(TimeCurrent())-
  StringReplace(filename, ":", "-"); // le caractère ":" n'est pas autorisé dans les n
//--- ouvre/crée la base de données dans le répertoire commun du terminal
  int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE|DATABASE_
  if(db==INVALID_HANDLE)
   {
    Print("Base de données : ", filename, " échec de l'ouverture, code d'erreur : ", G
    return;
   }
  else
    Print("Base de données : ", filename, " ouvert avec succès");

//--- vérifie si la table RATES existe
  if(DatabaseTableExists(db, "RATES"))
   {
    //--- supprime la table RATES
    if(!DatabaseExecute(db, "DROP TABLE IF EXISTS RATES"))
     {
      Print("Echec de la suppression de la table RATES, code d'erreur :", GetLastErro
      DatabaseClose(db);
      return;
     }
   }
//--- crée la table RATES
  if(!DatabaseExecute(db, "CREATE TABLE RATES("
                      "SYMBOL             CHAR(10),"
                      "TIME               INT NOT NULL,"
                      "OPEN               REAL,"
                      "HIGH               REAL,"
                      "LOW                REAL,"
                      "CLOSE              REAL,"
                      "TICK_VOLUME        INT,"
```

© 2000-2024, MetaQuotes Ltd.

```
                        "SPREAD             INT,"
                        "REAL_VOLUME        INT);"))
    {
     Print("DB: ", filename, " échec de la création de la table RATES, code d'erreur :'
     DatabaseClose(db);
     return;
    }
//--- affiche la liste de tous les champs de la table RATES
  if(DatabasePrint(db, "PRAGMA TABLE_INFO(RATES)", 0)<0)
    {
     PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(RATES)\") a échoué, code d'erreur=%
     DatabaseClose(db);
     return;
    }
//--- crée une requête paramétrée pour ajouter les barres à la table RATES
  string sql="INSERT INTO RATES (SYMBOL,TIME,OPEN,HIGH,LOW,CLOSE,TICK_VOLUME,SPREAD,RE
             " VALUES (?1,?2,?3,?4,?5,?6,?7,?8,?9)"; // paramètres de la requête
  int request=DatabasePrepare(db, sql);
  if(request==INVALID_HANDLE)
    {
     PrintFormat("DatabasePrepare() a échoué avec le code=%d", GetLastError());
     Print("Requête SQL : ", sql);
     DatabaseClose(db);
     return;
    }
//--- définit la valeur du premier paramètre de la requête
  DatabaseBind(request, 0, _Symbol);
//--- stocke l'heure de début avant d'ajouter des barres à la table RATES
  start=GetMicrosecondCount();
  DatabaseTransactionBegin(db);
  int total=ArraySize(rates);
  bool request_error=false;
  for(int i=0; i<total; i++)
    {
     //--- définit les valeurs des paramètres restants avant d'ajouter l'entrée
     ResetLastError();
     if(!DatabaseBind(request, 1, rates[i].time))
      {
       PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
       PrintFormat("Barre #%d ligne=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
     //--- si le dernier appel à DatabaseBind() a réussi, définit le paramètre suivant
     if(!request_error && !DatabaseBind(request, 2, rates[i].open))
      {
       PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
       PrintFormat("Barre #%d ligne=%d", i+1, __LINE__);
       request_error=true;
```

© 2000-2024, MetaQuotes Ltd.

```
      break;
     }
    if(!request_error && !DatabaseBind(request, 3, rates[i].high))
     {
      PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
      PrintFormat("Barre #%d ligne=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 4, rates[i].low))
     {
      PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
      PrintFormat("Barre #%d ligne=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 5, rates[i].close))
     {
      PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
      PrintFormat("Barre #%d ligne=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 6, rates[i].tick_volume))
     {
      PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
      PrintFormat("Barre #%d ligne=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 7, rates[i].spread))
     {
      PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
      PrintFormat("Barre #%d ligne=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 8, rates[i].real_volume))
     {
      PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
      PrintFormat("Barre #%d ligne=%d", i+1, __LINE__);
      request_error=true;
      break;
     }

    //--- exécute une requête pour insérer l'entrée et rechercher une erreur
    if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_MO
     {
      PrintFormat("DatabaseRead() a échoué avec le code=%d", GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

```
      DatabaseFinalize(request);
      request_error=true;
      break;
     }
    //--- réinitialise la requête avant la prochaine mise à jour des paramètres
    if(!request_error && !DatabaseReset(request))
     {
      PrintFormat("DatabaseReset() a échoué avec le code=%d", GetLastError());
      DatabaseFinalize(request);
      request_error=true;
      break;
     }
   } //--- fin du parcours toutes les barres

//--- état des transactions
  if(request_error)
   {
    PrintFormat("Table RATES : échec de l'ajout des %d barres ", ArraySize(rates));
    DatabaseTransactionRollback(db);
    DatabaseClose(db);
    return;
   }
  else
   {
    DatabaseTransactionCommit(db);
    PrintFormat("Table RATES : %d barres ajoutées en %d ms",
                ArraySize(rates), (GetMicrosecondCount()-start)/1000);
   }
//--- sauvegarde la table RATES dans un fichier CSV
  string csv_filename=Symbol()+".csv";
  long saved=DatabaseExport(db, "SELECT * FROM RATES", csv_filename, DATABASE_EXPORT_F
  if(saved>0)
    Print("Table RATES sauvegardée dans le fichier ", Symbol(), ".csv");
  else
    Print("DatabaseExport() a échoué. Erreur ", GetLastError());
//--- ferme la base de données
  DatabaseClose(db);
  PrintFormat("Base de données : %s créée et fermée avec succès", filename);
```

Voir aussi

DatabasePrint, DatabaseImport

© 2000-2024, MetaQuotes Ltd.

DatabasePrint

Ecrit une table ou le résultat de l'exécution d'une requête SQL dans le journal des Experts.

```
long  DatabasePrint(
   int           database,          // handle de la base de données obtenu avec Databa
   const string  table_or_sql,      // table ou requête SQL
   uint          flags              // combinaison de flags
   );
```

Paramètres

database

[in] Handle de la base de données obtenu avec DatabaseOpen().

table_or_sql

[in] Le nom d'une table ou le texte d'une requête SQL dont les résultats sont à écrire dans le journal des Experts.

flags

[in] Combinaison de flags déterminant le format de sortie. Les flags sont :

DATABASE_PRINT_NO_HEADER - n'affiche pas les noms des colonnes de la table (noms des champs)
DATABASE_PRINT_NO_INDEX - n'affiche pas les index des chaînes de caractères
DATABASE_PRINT_NO_FRAME - n'affiche pas de cadre séparant l'en-tête des données
DATABASE_PRINT_STRINGS_RIGHT - aligne les chaînes de caractères à droite.

Si flags=0, les colonnes et les chaînes de caractères sont affichées, l'en-tête et les données sont séparés par un cadre, et les chaînes de caractères sont alignées sur la gauche.

Valeur de Retour

Retourne le nombre de chaînes exportées ou -1 en cas d'erreur. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INTERNAL_ERROR (4001) - erreur critique d'exécution ;
- ERR_NOT_ENOUGH_MEMORY (4004) - mémoire insuffisante ;
- ERR_DATABASE_INTERNAL (5120) - erreur interne de la base de données ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données ;

Note

Si le journal doit afficher les résultats de la requête, la requête SQL doit commencer avec "SELECT" ou "select". En d'autres termes, la requête SQL ne peut pas modifier le statut de la base de données, sinon DatabasePrint() échouera avec une erreur.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   string filename="departments.sqlite";
//--- crée ou ouvre une base de données dans le répertoire commun du terminal
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATAB
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", l'ouverture du fichier filename, " a échouée avec le code ", GetLa
      return;
     }

//--- crée la table COMPANY
   if(!CreataTableCompany(db))
     {
      DatabaseClose(db);
      return;
     }
//--- crée la table DEPARTMENT
   if(!CreataTableDepartment(db))
     {
      DatabaseClose(db);
      return;
     }

//--- affiche la liste de tous les champs des tables COMPANY et DEPARTMENT
   PrintFormat("Essaye d'imprimer la requête \"PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)", 0)
     {
      PrintFormat("Echec de DatabasePrint(\"PRAGMA TABLE_INFO()\"), code d'erreur=%d",
      DatabaseClose(db);
      return;
     }
//--- affiche la table COMPANY dans le journal
   PrintFormat("Essaye d'imprimer la requête \"SELECT * from COMPANY\"");
   if(DatabasePrint(db, "SELECT * from COMPANY", 0)<0)
     {
      Print("Echec de DatabasePrint avec le code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- récupère le texte des tables COMPANY et DEPARTMENT
   string request="SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT "
                  "ON COMPANY.ID = DEPARTMENT.EMP_ID";
//--- affiche le résultat combiné
   PrintFormat("Essaye d'imprimer la requête \"SELECT EMP_ID, NAME, DEPT FROM COMPANY
   if(DatabasePrint(db, request, 0)<0)
     {
      Print("Echec de DatabasePrint avec le code ", GetLastError());
      DatabaseClose(db);
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- ferme la base de données
   DatabaseClose(db);
  }
/*
Conclusion :
Essaye d' imprimer la requête "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT
#| cid name    type     notnull dflt_value pk
-+-------------------------------------------
1|   0 ID      INT            1             1
2|   1 NAME    TEXT           1             0
3|   2 AGE     INT            1             0
4|   3 ADDRESS CHAR(50)       0             0
5|   4 SALARY  REAL           0             0
#| cid name   type     notnull dflt_value pk
-+-------------------------------------------
1|   0 ID     INT            1             1
2|   1 DEPT   CHAR(50)       1             0
3|   2 EMP_ID INT            1             0
Essaye d' imprimer la requête "SELECT * from COMPANY"
#| ID NAME  AGE ADDRESS     SALARY
-+--------------------------------
1|  1 Paul   32 California 25000.0
2|  2 Allen  25 Texas      15000.0
3|  3 Teddy  23 Norway     20000.0
4|  4 Mark   25 Rich-Mond  65000.0
5|  5 David  27 Texas      85000.0
6|  6 Kim    22 South-Hall 45000.0
7|  7 James  24 Houston    10000.0
Try to print request "SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMEN
#| EMP_ID NAME  DEPT
-+-------------------------
1|      1 Paul  IT Billing
2|      2 Allen Engineering
3|        Teddy
4|        Mark
5|        David
6|        Kim
7|      7 James Finance
*/
//+------------------------------------------------------------------+
//| Crée la table COMPANY                                            |
//+------------------------------------------------------------------+
bool CreateTableCompany(int database)
  {
//--- supprime la table COMPANY si elle existe
   if(DatabaseTableExists(database, "COMPANY"))
     {
      //--- supprime la table
      if(!DatabaseExecute(database, "DROP TABLE COMPANY"))
```

```
        {
         Print("Echec de la suppression de la table COMPANY avec le code ", GetLastEr
         return(false);
        }
     }
//--- crée la table COMPANY
   if(!DatabaseExecute(database, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
     {
      Print("DB: échec de création de la table COMPANY avec le code ", GetLastError())
      return(false);
     }

//--- insére les données dans la table COMPANY
   if(!DatabaseExecute(database, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VAI
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'A
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'T
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'M
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (5, 'D
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (6, 'P
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7, 'J
     {
      Print("Echec de l'insertion dans COMPANY avec le code ", GetLastError());
      return(false);
     }
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée la table DEPARTMENT                                         |
//+------------------------------------------------------------------+
bool CreateTableDepartment(int database)
  {
//--- supprime la table DEPARTMENT si elle existe
   if(DatabaseTableExists(database, "DEPARTMENT"))
     {
      //--- supprime la table
      if(!DatabaseExecute(database, "DROP TABLE DEPARTMENT"))
        {
         Print("Echec de la suppression de la table DEPARTMENT avec le code ", GetLast
         return(false);
        }
     }
//--- crée la table DEPARTMENT
   if(!DatabaseExecute(database, "CREATE TABLE DEPARTMENT ("
```

© 2000-2024, MetaQuotes Ltd.

```
                        "ID      INT PRIMARY KEY   NOT NULL,"
                        "DEPT    CHAR(50)          NOT NULL,"
                        "EMP_ID  INT               NOT NULL);"))
     {
      Print("DB: échec de création de la table DEPARTMENT avec le code ", GetLastErro
      return(false);
     }

//--- insére les données dans la table DEPARTMENT
   if(!DatabaseExecute(database, "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (1, '
                        "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (2, 'Engineerir
                        "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (3, 'Finance',
     {
      Print("Echec de l'insertion dans DEPARTMENT avec le code ", GetLastError());
      return(false);
     }
//--- succès
   return(true);
  }
//+------------------------------------------------------------------
```

Voir aussi

DatabaseExport, DatabaseImport

© 2000-2024, MetaQuotes Ltd.

DatabaseTableExists

Vérifie la présence d'une table dans la base de données.

```
bool  DatabaseTableExists(
   int     database,      // handle de la base de données retourné par DatabaseOpen
   string  table          // nom de la table
   );
```

Paramètres

database

[in] Handle de la base de données retourné par DatabaseOpen().

table

[in] Nom de la table.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INVALID_PARAMETER (4003) - aucun nom de table spécifié (chaîne vide ou NULL) ;
- ERR_WRONG_STRING_PARAMETER (5040) - erreur de conversion de la requête en UTF-8 ;
- ERR_DATABASE_INTERNAL (5120) - erreur interne de la base de données ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données ;
- ERR_DATABASE_EXECUTE (5124) - erreur d'exécution de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) - aucune table n'existe (ceci n'est pas une erreur, mais une fin normale).

Voir aussi

DatabasePrepare, DatabaseFinalize

© 2000-2024, MetaQuotes Ltd.

DatabaseExecute

Exécute une requête dans la base de données spécifiée.

```
bool  DatabaseExecute(
   int     database,      // handle de la base de données retourné par DatabaseOpen
   string  sql            // requête SQL
   );
```

Paramètres

database

[in] Handle de la base de données retourné par DatabaseOpen().

sql

[in] Requête SQL.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INTERNAL_ERROR (4001) - erreur critique d'exécution ;
- ERR_INVALID_PARAMETER (4003) - le paramètre sql contient une chaîne vide ;
- ERR_NOT_ENOUGH_MEMORY (4004) - mémoire insuffisante ;
- ERR_WRONG_STRING_PARAMETER (5040) - erreur de conversion de la requête en UTF-8 ;
- ERR_DATABASE_INTERNAL (5120) - erreur interne de la base de données ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données ;
- ERR_DATABASE_EXECUTE (5124) - erreur d'exécution de la requête.

Example:

```
//--- symbol statistics
struct Symbol_Stats
  {
   string            name;             // symbol name
   int               trades;           // number of trades for the symbol
   double            gross_profit;     // total profit for the symbol
   double            gross_loss;       // total loss for the symbol
   double            total_commission; // total commission for the symbol
   double            total_swap;       // total swaps for the symbol
   double            total_profit;     // total profit excluding swaps and commissions
   double            net_profit;       // net profit taking into account swaps and com
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swap
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
```

© 2000-2024, MetaQuotes Ltd.

```
  };

//--- Magic Number statistics
struct Magic_Stats
  {
   long              magic;            // EA's Magic Number
   int               trades;           // number of trades for the symbol
   double            gross_profit;     // total profit for the symbol
   double            gross_loss;       // total loss for the symbol
   double            total_commission; // total commission for the symbol
   double            total_swap;       // total swaps for the symbol
   double            total_profit;     // total profit excluding swaps and commissions
   double            net_profit;       // net profit taking into account swaps and com
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swap
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };

//--- entry hour statistics
struct Hour_Stats
  {
   char              hour_in;          // market entry hour
   int               trades;           // number of trades in this entry hour
   double            volume;           // volume of trades in this entry hour
   double            gross_profit;     // total profit in this entry hour
   double            gross_loss;       // total loss in this entry hour
   double            net_profit;       // net profit taking into account swaps and com
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swap
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };

int ExtDealsTotal=0;;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
```

```
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_stats.sqlite";
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
   PrintFormat("Deals in the trading history: %d ", ExtDealsTotal);

//--- get trading statistics per symbols
   int request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then (r.gross_profit+r.gro
                               "   (case when r.trades != 0 then r.win_trades*100.0/r.
                               "   (case when r.trades != 0 then r.loss_trades*100.0/r
                               "   r.gross_profit/r.win_trades as average_profit,"
                               "   r.gross_loss/r.loss_trades as average_loss,"
                               "   (case when r.gross_loss!=0.0 then r.gross_profit/(-
                               "FROM "
                               "   ("
                               "   SELECT SYMBOL,"
                               "   sum(case when entry =1 then 1 else 0 end) as trades
                               "   sum(case when profit > 0 then profit else 0 end) as
                               "   sum(case when profit < 0 then profit else 0 end) as
                               "   sum(swap) as total_swap,"
                               "   sum(commission) as total_commission,"
                               "   sum(profit) as total_profit,"
                               "   sum(profit+swap+commission) as net_profit,"
                               "   sum(case when profit > 0 then 1 else 0 end) as win_
                               "   sum(case when profit < 0 then 1 else 0 end) as loss
                               "   FROM DEALS "
                               "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                               "   GROUP BY SYMBOL"
                               "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Symbol_Stats stats[], symbol_stats;
   ArrayResize(stats, ExtDealsTotal);
   int i=0;
```

```
//--- get records from request results
   for(; DatabaseReadBind(request, symbol_stats) ; i++)
     {
      stats[i].name=symbol_stats.name;
      stats[i].trades=symbol_stats.trades;
      stats[i].gross_profit=symbol_stats.gross_profit;
      stats[i].gross_loss=symbol_stats.gross_loss;
      stats[i].total_commission=symbol_stats.total_commission;
      stats[i].total_swap=symbol_stats.total_swap;
      stats[i].total_profit=symbol_stats.total_profit;
      stats[i].net_profit=symbol_stats.net_profit;
      stats[i].win_trades=symbol_stats.win_trades;
      stats[i].loss_trades=symbol_stats.loss_trades;
      stats[i].expected_payoff=symbol_stats.expected_payoff;
      stats[i].win_percent=symbol_stats.win_percent;
      stats[i].loss_percent=symbol_stats.loss_percent;
      stats[i].average_profit=symbol_stats.average_profit;
      stats[i].average_loss=symbol_stats.average_loss;
      stats[i].profit_factor=symbol_stats.profit_factor;
     }
   ArrayResize(stats, i);
   Print("Trade statistics by Symbol");
   ArrayPrint(stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- get trading statistics for Expert Advisors by Magic Numbers
   request=DatabasePrepare(db, "SELECT r.*,"
                           "   (case when r.trades != 0 then (r.gross_profit+r.gross_l
                           "   (case when r.trades != 0 then r.win_trades*100.0/r.trac
                           "   (case when r.trades != 0 then r.loss_trades*100.0/r.tra
                           "   r.gross_profit/r.win_trades as average_profit,"
                           "   r.gross_loss/r.loss_trades as average_loss,"
                           "   (case when r.gross_loss!=0.0 then r.gross_profit/(-r.gr
                           "FROM "
                           "   ("
                           "   SELECT MAGIC,"
                           "   sum(case when entry =1 then 1 else 0 end) as trades,"
                           "   sum(case when profit > 0 then profit else 0 end) as gro
                           "   sum(case when profit < 0 then profit else 0 end) as gro
                           "   sum(swap) as total_swap,"
                           "   sum(commission) as total_commission,"
                           "   sum(profit) as total_profit,"
                           "   sum(profit+swap+commission) as net_profit,"
                           "   sum(case when profit > 0 then 1 else 0 end) as win_trac
                           "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                           "   FROM DEALS "
                           "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
```

© 2000-2024, MetaQuotes Ltd.

```
                              "   GROUP BY MAGIC"
                              "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Magic_Stats EA_stats[], magic_stats;
   ArrayResize(EA_stats, ExtDealsTotal);
   i=0;
//--- print
   for(; DatabaseReadBind(request, magic_stats) ; i++)
     {
      EA_stats[i].magic=magic_stats.magic;
      EA_stats[i].trades=magic_stats.trades;
      EA_stats[i].gross_profit=magic_stats.gross_profit;
      EA_stats[i].gross_loss=magic_stats.gross_loss;
      EA_stats[i].total_commission=magic_stats.total_commission;
      EA_stats[i].total_swap=magic_stats.total_swap;
      EA_stats[i].total_profit=magic_stats.total_profit;
      EA_stats[i].net_profit=magic_stats.net_profit;
      EA_stats[i].win_trades=magic_stats.win_trades;
      EA_stats[i].loss_trades=magic_stats.loss_trades;
      EA_stats[i].expected_payoff=magic_stats.expected_payoff;
      EA_stats[i].win_percent=magic_stats.win_percent;
      EA_stats[i].loss_percent=magic_stats.loss_percent;
      EA_stats[i].average_profit=magic_stats.average_profit;
      EA_stats[i].average_loss=magic_stats.average_loss;
      EA_stats[i].profit_factor=magic_stats.profit_factor;
     }
   ArrayResize(EA_stats, i);
   Print("Trade statistics by Magic Number");
   ArrayPrint(EA_stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- make sure that hedging system for open position management is used on the accour
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
     {
      //--- deals cannot be transformed to trades using a simple method through transa
      DatabaseClose(db);
      return;
     }

//--- now create the TRADES table based on the DEALS table
   if(!CreateTableTrades(db))
     {
```

```
      DatabaseClose(db);
      return;
     }
//--- fill in the TRADES table using an SQL query based on DEALS table data
   if(DatabaseTableExists(db, "DEALS"))
      //--- populate the TRADES table
      if(!DatabaseExecute(db, "INSERT INTO TRADES(TIME_IN,HOUR_IN,TICKET,TYPE,VOLUME,S
                          "SELECT "
                          "   d1.time as time_in,"
                          "   d1.hour as hour_in,"
                          "   d1.position_id as ticket,"
                          "   d1.type as type,"
                          "   d1.volume as volume,"
                          "   d1.symbol as symbol,"
                          "   d1.price as price_in,"
                          "   d2.time as time_out,"
                          "   d2.price as price_out,"
                          "   d1.commission+d2.commission as commission,"
                          "   d2.swap as swap,"
                          "   d2.profit as profit "
                          "FROM DEALS d1 "
                          "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                          "WHERE d1.entry=0 AND d2.entry=1     "))
        {
         Print("DB: fillng the table TRADES failed with code ", GetLastError());
         return;
        }

//--- get trading statistics by market entry hours
   request=DatabasePrepare(db, "SELECT r.*,"
                           "   (case when r.trades != 0 then (r.gross_profit+r.gross_l
                           "   (case when r.trades != 0 then r.win_trades*100.0/r.trac
                           "   (case when r.trades != 0 then r.loss_trades*100.0/r.tra
                           "   r.gross_profit/r.win_trades as average_profit,"
                           "   r.gross_loss/r.loss_trades as average_loss,"
                           "   (case when r.gross_loss!=0.0 then r.gross_profit/(-r.gr
                           "FROM "
                           "   ("
                           "   SELECT HOUR_IN,"
                           "   count() as trades,"
                           "   sum(volume) as volume,"
                           "   sum(case when profit > 0 then profit else 0 end) as gro
                           "   sum(case when profit < 0 then profit else 0 end) as gro
                           "   sum(profit) as net_profit,"
                           "   sum(case when profit > 0 then 1 else 0 end) as win_trac
                           "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                           "   FROM TRADES "
                           "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                           "   GROUP BY HOUR_IN"
```

© 2000-2024, MetaQuotes Ltd.

```
                              "    ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Hour_Stats hours_stats[], h_stats;
   ArrayResize(hours_stats, ExtDealsTotal);
   i=0;
//--- print
   for(; DatabaseReadBind(request, h_stats) ; i++)
     {
      hours_stats[i].hour_in=h_stats.hour_in;
      hours_stats[i].trades=h_stats.trades;
      hours_stats[i].volume=h_stats.volume;
      hours_stats[i].gross_profit=h_stats.gross_profit;
      hours_stats[i].gross_loss=h_stats.gross_loss;
      hours_stats[i].net_profit=h_stats.net_profit;
      hours_stats[i].win_trades=h_stats.win_trades;
      hours_stats[i].loss_trades=h_stats.loss_trades;
      hours_stats[i].expected_payoff=h_stats.expected_payoff;
      hours_stats[i].win_percent=h_stats.win_percent;
      hours_stats[i].loss_percent=h_stats.loss_percent;
      hours_stats[i].average_profit=h_stats.average_profit;
      hours_stats[i].average_loss=h_stats.average_loss;
      hours_stats[i].profit_factor=h_stats.profit_factor;
     }
   ArrayResize(hours_stats, i);
   Print("Trade statistics by entry hour");
   ArrayPrint(hours_stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- close database
   DatabaseClose(db);
   return;
  }
/*
Deals in the trading history: 2771
Trade statistics by Symbol
      [name] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap] [tot
[0] "AUDUSD"      112      503.20000   -568.00000           -8.83000    -24.64000
[1] "EURCHF"      125      607.71000   -956.85000          -11.77000    -45.02000
[2] "EURJPY"      127     1078.49000  -1057.83000          -10.61000    -45.76000
[3] "EURUSD"      233     1685.60000  -1386.80000          -41.00000    -83.76000
[4] "GBPCHF"      125     1881.37000  -1424.72000          -22.60000    -51.56000
[5] "GBPJPY"      127     1943.43000  -1776.67000          -18.84000    -52.46000
```

© 2000-2024, MetaQuotes Ltd.

```
[6] "GBPUSD"      121     1668.50000  -1438.20000          -7.96000    -49.93000
[7] "USDCAD"       99      405.28000   -475.47000          -8.68000    -31.68000
[8] "USDCHF"      206     1588.32000  -1241.83000         -17.98000    -65.92000
[9] "USDJPY"      107      464.73000   -730.64000         -35.12000    -34.24000

Trade statistics by Magic Number
    [magic] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap] [tota
[0]     100      242     2584.80000  -2110.00000          -33.36000    -93.53000
[1]     200      254     3021.92000  -2834.50000          -29.45000    -98.22000
[2]     300      250     2489.08000  -2381.57000          -34.37000    -96.58000
[3]     400      224     1272.50000  -1283.00000          -24.43000    -64.80000
[4]     500      198     1141.23000  -1051.91000          -27.66000    -63.36000
[5]     600      214     1317.10000  -1396.03000          -34.12000    -68.48000

Trade statistics by entry hour
     [hour_in] [trades] [volume] [gross_profit] [gross_loss] [net_profit] [win_trades]
[ 0]         0       50  5.00000      336.51000   -747.47000   -410.96000           2
[ 1]         1       20  2.00000      102.56000    -57.20000     45.36000           1
[ 2]         2        6  0.60000       38.55000    -14.60000     23.95000           5
[ 3]         3       38  3.80000      173.84000   -200.15000    -26.31000           2
[ 4]         4       60  6.00000      361.44000   -389.40000    -27.96000           2
[ 5]         5       32  3.20000      157.43000   -179.89000    -22.46000           2
[ 6]         6       18  1.80000       95.59000   -162.33000    -66.74000           1
[ 7]         7       14  1.40000       38.48000   -134.30000    -95.82000           9
[ 8]         8       42  4.20000      368.48000   -322.30000     46.18000           2
[ 9]         9      118 11.80000     1121.62000   -875.21000    246.41000           7
[10]        10      206 20.60000     2280.59000  -2021.80000    258.79000          11
[11]        11      138 13.80000     1377.02000   -994.18000    382.84000           8
[12]        12      152 15.20000     1247.56000  -1463.80000   -216.24000           8
[13]        13       64  6.40000      778.27000   -516.22000    262.05000           3
[14]        14       62  6.20000      536.93000   -427.47000    109.46000           3
[15]        15       50  5.00000      699.92000   -413.00000    286.92000           2
[16]        16       88  8.80000      778.55000   -514.00000    264.55000           5
[17]        17       76  7.60000      533.92000  -1019.46000   -485.54000           4
[18]        18       52  5.20000      237.17000   -246.78000     -9.61000           2
[19]        19       52  5.20000      407.67000   -150.36000    257.31000           3
[20]        20       18  1.80000       65.92000    -89.09000    -23.17000           9
[21]        21       10  1.00000       41.86000    -32.38000      9.48000           7
[22]        22       14  1.40000       45.55000    -83.72000    -38.17000           6
[23]        23        2  0.20000        1.20000     -1.90000     -0.70000           1
*/

//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(database, "DEALS"))
```

```
      {
       return(false);
      }
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "HOUR        INT,"
                          "REASON      INT);"))
        {
         Print("DB: create the DEALS table  failed with code ", GetLastError());
         return(false);
        }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   ExtDealsTotal=HistoryDealsTotal();
//--- add deals to the table
   if(!InsertDeals(database))
      return(false);
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop the DEALS table with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
```

© 2000-2024, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was exec
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number (Expert Advisor ID)
   long     reason;              // deal execution reason or source
   char     hour;                // deal execution hour
   MqlDateTime time_strusture;
//--- go through all deals and add them to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- lock the database before executing transactions
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      TimeToStruct(time, time_strusture);
      hour= (char)time_strusture.hour;
      //--- add each deal to the table using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
```

```
                                        "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                        deal_ticket, order_ticket, position_ticket, tir
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%d with code %d", __FUNCTION__, deal_
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- roll back all transactions and unlock the database
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastErro
      return(false);
     }
//--- all transactions have been performed successfully - record changes and unlock th
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the TRADES table                                         |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- if the TRADES table already exists, delete it
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- check if the table exists
   if(!DatabaseTableExists(database, "TRADES"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "HOUR_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
         Print("DB: create the TRADES table failed with code ", GetLastError());
         return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
```

Voir aussi

DatabasePrepare, DatabaseFinalize

© 2000-2024, MetaQuotes Ltd.

DatabasePrepare

Crée le handle d'une requête, qui peut être ensuite exécutée avec DatabaseRead().

```
int  DatabasePrepare(
   int     database,       // handle de la base de données retourné par DatabaseOpen
   string  sql,            // requête SQL
           ...             // paramètres de la requête
   );
```

Paramètres

database

[in] Handle de la base de données retourné par DatabaseOpen().

sql

[in] La requête SQL pouvant contenir des paramètres substitués automatiquement et nommés ?1,?2,...

...

[in] Paramètres de la requête substitués automatiquement.

Valeur de Retour

La fonction retourne un handle pour la requête SQL en cas de succès. Sinon, elle retourne INVALID_HANDLE. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INVALID_PARAMETER (4003) - chemin vers le fichier de base de données contenant une chaîne vide, ou la combinaison de flag est invalide ;
- ERR_NOT_ENOUGH_MEMORY (4004) - mémoire insuffisante ;
- ERR_WRONG_STRING_PARAMETER (5040) - erreur de conversion de la requête en UTF-8 ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données ;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - le nombre d'objets Base de données est supérieur au maximum autorisé ;
- ERR_DATABASE_PREPARE (5125) - erreur de génération de la requête.

Note

La fonction DatabasePrepare() n'effectue aucune requête sur la base de données. Son but est de vérifier les paramètres de la requête et de retourner le handle pour exécuter la requête SQL suivant les résultats de la vérification. La requête elle-même est définie lors du premier appel à DatabaseRead().

Example:

```
//--- Structure to store the deal
struct Deal
  {
   ulong             ticket;           // DEAL_TICKET
   long              order_ticket;     // DEAL_ORDER
   long              position_ticket;  // DEAL_POSITION_ID
   datetime          time;             // DEAL_TIME
   char              type;             // DEAL_TYPE
```

© 2000-2024, MetaQuotes Ltd.

```
   char              entry;            // DEAL_ENTRY
   string            symbol;           // DEAL_SYMBOL
   double            volume;           // DEAL_VOLUME
   double            price;            // DEAL_PRICE
   double            profit;           // DEAL_PROFIT
   double            swap;             // DEAL_SWAP
   double            commission;       // DEAL_COMMISSION
   long              magic;            // DEAL_MAGIC
   char              reason;           // DEAL_REASON
  };
//--- Structure to store the trade: the order of members corresponds to the position i
struct Trade
  {
   datetime          time_in;          // entry time
   ulong             ticket;           // position ID
   char              type;             // buy or sell
   double            volume;           // volume
   string            symbol;           // symbol
   double            price_in;         // entry price
   datetime          time_out;         // exit time
   double            price_out;        // exit price
   double            commission;       // entry and exit commission
   double            swap;             // swap
   double            profit;           // profit or loss
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_trades.sqlite"
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
```

© 2000-2024, MetaQuotes Ltd.

```
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("Deals in the trading history: %d ", deals_total);
//--- add deals to the table
   if(!InsertDeals(db))
      return;
//--- show the first 10 deals
   Deal deals[], deal;
   ArrayResize(deals, 10);
   int request=DatabasePrepare(db, "SELECT * FROM DEALS");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   int i;
   for(i=0; DatabaseReadBind(request, deal); i++)
     {
      if(i>=10)
         break;
      deals[i].ticket=deal.ticket;
      deals[i].order_ticket=deal.order_ticket;
      deals[i].position_ticket=deal.position_ticket;
      deals[i].time=deal.time;
      deals[i].type=deal.type;
      deals[i].entry=deal.entry;
      deals[i].symbol=deal.symbol;
      deals[i].volume=deal.volume;
      deals[i].price=deal.price;
      deals[i].profit=deal.profit;
      deals[i].swap=deal.swap;
      deals[i].commission=deal.commission;
      deals[i].magic=deal.magic;
      deals[i].reason=deal.reason;
     }
//--- print the deals
   if(i>0)
     {
      ArrayResize(deals, i);
      PrintFormat("The first %d deals:", i);
      ArrayPrint(deals);
     }

//--- delete request after use
   DatabaseFinalize(request);

//--- make sure that hedging system for open position management is used on the accour
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- deals cannot be transformed to trades using a simple method through transa
       DatabaseClose(db);
       return;
      }

//--- now create the TRADES table based on the DEALS table
   if(!CreateTableTrades(db))
      {
       DatabaseClose(db);
       return;
      }
//--- fill in the TRADES table using an SQL query based on DEALS table data
   ulong start=GetMicrosecondCount();
   if(DatabaseTableExists(db, "DEALS"))
      //--- populate the TRADES table
      if(!DatabaseExecute(db, "INSERT INTO TRADES(TIME_IN,TICKET,TYPE,VOLUME,SYMBOL,PR
                          "SELECT "
                          "   d1.time as time_in,"
                          "   d1.position_id as ticket,"
                          "   d1.type as type,"
                          "   d1.volume as volume,"
                          "   d1.symbol as symbol,"
                          "   d1.price as price_in,"
                          "   d2.time as time_out,"
                          "   d2.price as price_out,"
                          "   d1.commission+d2.commission as commission,"
                          "   d2.swap as swap,"
                          "   d2.profit as profit "
                          "FROM DEALS d1 "
                          "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                          "WHERE d1.entry=0 AND d2.entry=1     "))
         {
          Print("DB: fillng the TRADES table failed with code ", GetLastError());
          return;
         }
   ulong transaction_time=GetMicrosecondCount()-start;

//--- show the first 10 deals
   Trade trades[], trade;
   ArrayResize(trades, 10);
   request=DatabasePrepare(db, "SELECT * FROM TRADES");
   if(request==INVALID_HANDLE)
      {
       Print("DB: ", filename, " request failed with code ", GetLastError());
       DatabaseClose(db);
       return;
      }
   for(i=0; DatabaseReadBind(request, trade); i++)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       if(i>=10)
          break;
       trades[i].time_in=trade.time_in;
       trades[i].ticket=trade.ticket;
       trades[i].type=trade.type;
       trades[i].volume=trade.volume;
       trades[i].symbol=trade.symbol;
       trades[i].price_in=trade.price_in;
       trades[i].time_out=trade.time_out;
       trades[i].price_out=trade.price_out;
       trades[i].commission=trade.commission;
       trades[i].swap=trade.swap;
       trades[i].profit=trade.profit;
      }
//--- print trades
   if(i>0)
      {
       ArrayResize(trades, i);
       PrintFormat("\r\nThe first %d trades:", i);
       ArrayPrint(trades);
       PrintFormat("Filling the TRADES table took %.2f milliseconds",double(transactior
      }
//--- delete request after use
   DatabaseFinalize(request);

//--- close the database
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2741
   The first 10 deals:
       [ticket] [order_ticket] [position_ticket]              [time] [type] [entry] [s
   [0] 34429573              0                 0 2019.09.05 22:39:59      2       0 ""
   [1] 34432127       51447238          51447238 2019.09.06 06:00:03      0       0 "U
   [2] 34432128       51447239          51447239 2019.09.06 06:00:03      1       0 "U
   [3] 34432450       51447565          51447565 2019.09.06 07:00:00      0       0 "E
   [4] 34432456       51447571          51447571 2019.09.06 07:00:00      1       0 "A
   [5] 34432879       51448053          51448053 2019.09.06 08:00:00      1       0 "U
   [6] 34432888       51448064          51448064 2019.09.06 08:00:00      0       0 "U
   [7] 34435147       51450470          51450470 2019.09.06 10:30:00      1       0 "E
   [8] 34435152       51450476          51450476 2019.09.06 10:30:00      0       0 "C
   [9] 34435154       51450479          51450479 2019.09.06 10:30:00      1       0 "E

   The first 10 trades:
                  [time_in] [ticket] [type] [volume] [symbol] [price_in]          [time
   [0] 2019.09.06 06:00:03 51447238      0  0.10000 "USDCAD"    1.32320 2019.09.06 18:
   [1] 2019.09.06 06:00:03 51447239      1  0.10000 "USDCHF"    0.98697 2019.09.06 18:
```

© 2000-2024, MetaQuotes Ltd.

```
   [2] 2019.09.06 07:00:00 51447565      0  0.10000 "EURUSD"    1.10348 2019.09.09 03:
   [3] 2019.09.06 07:00:00 51447571      1  0.10000 "AUDUSD"    0.68203 2019.09.09 03:
   [4] 2019.09.06 08:00:00 51448053      1  0.10000 "USDCHF"    0.98701 2019.09.06 18:
   [5] 2019.09.06 08:00:00 51448064      0  0.10000 "USDJPY"  106.96200 2019.09.06 18:
   [6] 2019.09.06 10:30:00 51450470      1  0.10000 "EURUSD"    1.10399 2019.09.06 14:
   [7] 2019.09.06 10:30:00 51450476      0  0.10000 "GBPUSD"    1.23038 2019.09.06 14:
   [8] 2019.09.06 10:30:00 51450479      1  0.10000 "EURJPY"  118.12000 2019.09.06 14:
   [9] 2019.09.06 10:30:00 51450480      0  0.10000 "GBPJPY"  131.65300 2019.09.06 14:
   Filling the TRADES table took 12.51 milliseconds
*/
//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(database, "DEALS"))
     {
      return(false);
     }
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
        {
         Print("DB: create the DEALS table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
```

```
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop the DEALS table  with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was exec
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number (Expert Advisor ID)
   long     reason;              // deal execution reason or source
//--- go through all deals and add them to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- lock the database before executing transactions
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
```

© 2000-2024, MetaQuotes Ltd.

```
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- add each deal to the table using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                       deal_ticket, order_ticket, position_ticket, tim
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%d with code %d", __FUNCTION__, deal_
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- roll back all transactions and unlock the database
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code %d", __FUNCTION__, GetLastEr
      return(false);
     }
//--- all transactions have been performed successfully - record changes and unlock th
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the TRADES table                                         |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- if the TRADES table already exists, delete it
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- check if the table exists
   if(!DatabaseTableExists(database, "TRADES"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
```

```
         Print("DB: create the TRADES table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
```

Voir aussi

DatabaseExecute, DatabaseFinalize

© 2000-2024, MetaQuotes Ltd.

DatabaseReset

Réinitialise une requête, comme après avoir appelé DatabasePrepare().

```
int  DatabaseReset(
   int  request      // handle de requête récupéré de DatabasePrepare
   );
```

Paramètres

request

[in] Le handle de la requête obtenu avec DatabasePrepare().

Valeur de Retour

Retourne true en cas de succès, sinon - false. Pour obtenir le code de l'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_DATABASE_INVALID_HANDLE (5121) - handle de base de données invalide ;
- Des codes d'erreur SQLite commençant avec ERR_DATABASE_ERROR(5601).

Note

La fonction DatabaseReset() est destinée à l'exécution multiple d'une requête avec différentes valeurs de paramètres. Par exemple, lors de l'ajout de données à la table en bloc à l'aide de la commande INSERT, un ensemble personnalisé de valeurs de champ doit être formé pour chaque entrée.

Contrairement à DatabasePrepare(), l'appel DatabaseReset() ne compile pas la chaîne avec les commandes SQL dans une nouvelle requête, donc DatabaseReset() est exécuté beaucoup plus rapidement que DatabasePrepare().

DatabaseReset() est utilisé avec les fonctions DatabaseBind() et/ou DatabaseBindArray() si les valeurs des paramètres de la requête doivent être modifiées après l'exécution de DatabaseRead(). Cela signifie qu'avant de définir de nouvelles valeurs des paramètres de requête (avant le bloc d'appels DatabaseBind/DatabaseBindArray), DatabaseReset() doit être appelé pour le réinitialiser. La requête paramétrée elle-même doit être créée à l'aide de DatabasePrepare().

Tout comme DatabasePrepare(), DatabaseReset() ne fait pas de requête en base de données. Une exécution directe de la requête est effectuée lors de l'appel de DatabaseRead() ou DatabaseReadBind().

L'appel DatabaseReset() ne conduit pas à réinitialiser les valeurs des paramètres dans la requête si elles ont été définies en appelant DatabaseBind()/DatabaseBindArray(), c'est-à-dire que les paramètres conservent leurs valeurs. Ainsi, la valeur d'un seul paramètre peut être modifiée. Il n'est pas nécessaire de définir à nouveau tous les paramètres de demande après avoir appelé DatabaseReset().

Le handle d'une requête supprimée à l'aide de DatabaseFinalize() ne peut pas être transmis à DatabaseReset(). Cela entraînera une erreur.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- crée ou ouvre une base de données
   string filename="symbols.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
 Print("DB: échec de l'ouverture de ", filename,  avec le code ", GetLastError());
      return;
     }
   else
 Print("Base de données :", filename, " ouvert avec succès");
//--- supprime la table SYMBOLS si elle existe
   if(DatabaseTableExists(db, "SYMBOLS"))
     {
      //--- supprime la table
      if(!DatabaseExecute(db, "DROP TABLE SYMBOLS"))
        {
         Print("Impossible de supprimer la table SYMBOLS avec le code ", GetLastError
         DatabaseClose(db);
         return;
        }
     }
//--- crée la table SYMBOLS
   if(!DatabaseExecute(db, "CREATE TABLE SYMBOLS("
                       "NAME           TEXT    NOT NULL,"
                       "DESCRIPTION    TEXT            ,"
                       "PATH           TEXT            ,"
                       "SPREAD         INT             ,"
                       "POINT          REAL    NOT NULL,"
                       "DIGITS         INT     NOT NULL,"
                       "JSON           BLOB );"))
     {
      Print("DB : ", filename, " échec de 'create table' avec le code ", GetLastError
      DatabaseClose(db);
      return;
     }
//--- affiche la liste de tous les champs de la table TICKS
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(SYMBOLS)", 0)<0)
     {
      PrintFormat("Echec de DatabasePrint(\"PRAGMA TABLE_INFO(SYMBOLS)\"), code d'erre
      DatabaseClose(db);
      return;
     }

//--- crée une requête paramétrée pour ajouter des symboles à la table SYMBOLS
   string sql="INSERT INTO SYMBOLS (NAME,DESCRIPTION,PATH,SPREAD,POINT,DIGITS,JSON)"
              " VALUES (?1,?2,?3,?4,?5,?6,?7);"; // paramètres de la requête
   int request=DatabasePrepare(db, sql);
```

© 2000-2024, MetaQuotes Ltd.

```
   if(request==INVALID_HANDLE)
     {
 PrintFormat("DatabasePrepare() a échoué avec le code=%d", GetLastError());
      Print("Requête SQL : ", sql);
      DatabaseClose(db);
      return;
     }

//--- parcours tous les symboles et les ajoute à la table SYMBOLS
   int symbols=SymbolsTotal(false);
   bool request_error=false;
   DatabaseTransactionBegin(db);
   for(int i=0; i<symbols; i++)
     {
      //--- définit les valeurs des paramètres avant d'ajouter un symbole
      ResetLastError();
      string symbol=SymbolName(i, false);
      if(!DatabaseBind(request, 0, symbol))
        {
         PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LINE__
         request_error=true;
         break;
        }
      //--- si l'appel précédent DatabaseBind () a réussi, définit le paramètre suivan
      if(!DatabaseBind(request, 1, SymbolInfoString(symbol, SYMBOL_DESCRIPTION)))
        {
         PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LINE__
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 2, SymbolInfoString(symbol, SYMBOL_PATH)))
        {
         PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LINE__
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 3, SymbolInfoInteger(symbol, SYMBOL_SPREAD)))
        {
         PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LINE__
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 4, SymbolInfoDouble(symbol, SYMBOL_POINT)))
        {
         PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LINE__
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 5, SymbolInfoInteger(symbol, SYMBOL_DIGITS)))
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LINE__
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 6, GetSymBolAsJson(symbol)))
        {
         PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LINE__
         request_error=true;
         break;
        }

 //--- exécute une requête d'insertion de l'entrée et recherche l'erreur
      if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
        {
 PrintFormat("DatabaseRead() a échoué avec le code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
      else
         PrintFormat("%d: %s ajouté", i+1, symbol);
 //--- réinitialise la requête avant la prochaine mise à jour des paramètres
      if(!DatabaseReset(request))
        {
 PrintFormat("DatabaseReset() a échoué avec le code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
     } //--- tous les symboles ont été parcourus

//--- statuts des transactions
   if(request_error)
     {
      PrintFormat("Table SYMBOLS : échec de l'ajout de %d symboles", symbols);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table SYMBOLS : %d symboles ajoutés",symbols);
     }

//--- sauvegarde la table SYMBOLS dans un fichier CSV
   string csv_filename="symbols.csv";
   if(DatabaseExport(db, "SELECT * FROM SYMBOLS", csv_filename,
```

© 2000-2024, MetaQuotes Ltd.

```
                        DATABASE_EXPORT_HEADER|DATABASE_EXPORT_INDEX|DATABASE_EXPORT_QUOT
      Print("Base de données : la table SYMBOLS a été sauvée dans ", csv_filename);
   else
      Print("Base de données : DatabaseExport(\"SELECT * FROM SYMBOLS\") a échoué avec

//--- ferme le fichier de base de données et information
   DatabaseClose(db);
 PrintFormat("Base de données : %s créée et fermée", filename);
  }
//+------------------------------------------------------------------+
//| Retourne la spécification d'un symbole au format JSON            |
//+------------------------------------------------------------------+
string GetSymBolAsJson(string symbol)
  {
//--- indents
   string indent1=Indent(1);
   string indent2=Indent(2);
   string indent3=Indent(3);
//---
   int digits=(int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   string json="{"+
               "\n"+indent1+"\"ConfigSymbols\":["+
               "\n"+indent2+"{"+
               "\n"+indent3+"\"Symbol\":\""+symbol+"\","+
               "\n"+indent3+"\"Path\":\""+SymbolInfoString(symbol, SYMBOL_PATH)+"\","+
               "\n"+indent3+"\"CurrencyBase\":\""+SymbolInfoString(symbol, SYMBOL_CURR
               "\n"+indent3+"\"CurrencyProfit\":\""+SymbolInfoString(symbol, SYMBOL_CU
               "\n"+indent3+"\"CurrencyMargin\":\""+SymbolInfoString(symbol, SYMBOL_CU
               "\n"+indent3+"\"ColorBackground\":\""+ColorToString((color)SymbolInfoIn
               "\n"+indent3+"\"Digits\":\""+IntegerToString(SymbolInfoInteger(symbol,
               "\n"+indent3+"\"Point\":\""+DoubleToString(SymbolInfoDouble(symbol, SYM
               "\n"+indent3+"\"TickBookDepth\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"ChartMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"TradeMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"TradeCalcMode\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"OrderMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"CalculationMode\":\""+IntegerToString(SymbolInfoInteger
               "\n"+indent3+"\"ExecutionMode\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"ExpirationMode\":\""+IntegerToString(SymbolInfoInteger
               "\n"+indent3+"\"FillFlags\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"ExpirFlags\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"Spread\":\""+IntegerToString(SymbolInfoInteger(symbol,
               "\n"+indent3+"\"TickValue\":\""+StringFormat("%G", (SymbolInfoDouble(sy
               "\n"+indent3+"\"TickSize\":\""+StringFormat("%G", (SymbolInfoDouble(sym
               "\n"+indent3+"\"ContractSize\":\""+StringFormat("%G",(SymbolInfoDouble
               "\n"+indent3+"\"StopsLevel\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"VolumeMin\":\""+StringFormat("%G",(SymbolInfoDouble(sym
               "\n"+indent3+"\"VolumeMax\":\""+StringFormat("%G",(SymbolInfoDouble(sym
               "\n"+indent3+"\"VolumeStep\":\""+StringFormat("%G",(SymbolInfoDouble(sy
```

© 2000-2024, MetaQuotes Ltd.

```
                "\n"+indent3+"\"VolumeLimit\":\""+StringFormat("%G",(SymbolInfoDouble(s
                "\n"+indent3+"\"SwapMode\":\""+IntegerToString(SymbolInfoInteger(symbol
                "\n"+indent3+"\"SwapLong\":\""+StringFormat("%G",(SymbolInfoDouble(symb
                "\n"+indent3+"\"SwapShort\":\""+StringFormat("%G",(SymbolInfoDouble(sym
                "\n"+indent3+"\"Swap3Day\":\""+IntegerToString(SymbolInfoInteger(symbol
                "\n"+indent2+"}"+
                "\n"+indent1+"]"+
                "\n}";
   return(json);
  }
//+------------------------------------------------------------------+
//| Forme une indentation composée d'espaces                         |
//+------------------------------------------------------------------+
string Indent(const int number, const int characters=3)
  {
   int length=number*characters;
   string indent=NULL;
   StringInit(indent, length, ' ');
   return indent;
  }
/*
 Résultat :
 Base de données : symbols.sqlite ouverte avec succès
  #| cid name        type notnull dflt_value pk
  -+-----------------------------------------
  1|   0 NAME        TEXT       1             0
  2|   1 DESCRIPTION TEXT       0             0
  3|   2 PATH        TEXT       0             0
  4|   3 SPREAD      INT        0             0
  5|   4 POINT       REAL       1             0
  6|   5 DIGITS      INT        1             0
  7|   6 JSON        BLOB       0             0
 1: EURUSD ajouté
 2: GBPUSD ajouté
 3: USDCHF ajouté
  ...
 82: USDCOP ajouté
 83: USDARS ajouté
 84: USDCLP ajouté
 Table SYMBOLS : 84 symboles ajoutés
 Base de données : table SYMBOLS sauvée dans symbols.csv
 Base de données : symbols.sqlite créée et fermée
*/
```

Voir aussi

DatabasePrepare, DatabaseBind, DatabaseBindArray, DatabaseFinalize

© 2000-2024, MetaQuotes Ltd.

DatabaseBind

Définit la valeur d'un paramètre dans une requête.

```
bool  DatabaseBind(
   int  request,       // le handle d'une requête créée avec DatabasePrepare
   int  index,         // l'indice du paramètre dans la requête
   T    value          // la valeur d'un paramètre de type simple
   );
```

Paramètres

request

[in] Le handle d'une requête créée avec DatabasePrepare().

index

[in] L'indice du paramètre dans la requête pour lequel la valeur doit être définie. La numérotation commence à zéro.

value

[in] La valeur à définir. Types étendus : bool, char, uchar, short, ushart, int, uint, color, datetime, long, ulong, float, double, string.

Valeur de Retour

Retourne true en cas de succès, sinon - false. Pour obtenir le code de l'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INVALID_PARAMETER (4003) - type non supporté ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle de base de données invalide ;
- ERR_DATABASE_NOT_READY (5128) - ne peut pas utiliser actuellement la fonction pour effectuer une requpete. La requête est en cours d'exécution ou déjà terminée. DatabaseReset() doit être appelé.

Note

La fonction est utilisée dans le cas où une requête SQL contient "?" ou "?N" valeurs paramétrables où N signifie l'indice du paramètre (à partir de un). Dans le même temps, l'indexation des paramètres dans DatabaseBind () commence à zéro.

Par exemple :

```
INSERT INTO table VALUES (?,?,?)
SELECT * FROM table WHERE id=?
```

La fonction peut être appelée immédiatement après la création d'une demande paramétrée dans DatabasePrepare() ou après la réinitialisation de la demande à l'aide de DatabaseReset () .

Utilisez cette fonction avec DatabaseReset() pour exécuter la requête autant de fois que nécessaire avec différentes valeurs de paramètre.

La fonction est conçue pour fonctionner avec des paramètres de type simples. Si un paramètre doit être vérifié par rapport à un tableau, utilisez la fonction DatabaseBindArray().

© 2000-2024, MetaQuotes Ltd.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlTick ticks[];
//--- mémorise l'heure de début avant de recevoir les ticks
   uint start=GetTickCount();
//--- demande l'historique des ticks par jour
   ulong to=TimeCurrent()*1000;
   ulong from=to-PeriodSeconds(PERIOD_D1)*1000;
   if(CopyTicksRange(_Symbol, ticks, COPY_TICKS_ALL, from, to)==-1)
     {
 PrintFormat("%s : CopyTicksRange (%s - %s) a échoué, erreur=%d",
                  _Symbol, TimeToString(datetime(from/1000)), TimeToString(datetime(to
      return;
     }
   else
     {
 //--- combien de ticks ont été reçus et combien de temps il a fallu pour les recevoir
 PrintFormat("%s : CopyTicksRange a reçu %d ticks en %d ms (de %s à %s)",
                  _Symbol, ArraySize(ticks), GetTickCount()-start,
                  TimeToString(datetime(from/1000)), TimeToString(datetime(to/1000)));
     }

//--- définit le nom du fichier pour stocker la base de données
   string filename=_Symbol+" "+TimeToString(datetime(from/1000))+" - "+TimeToString(da
 StringReplace(nom de fichier, ": ", "."); // le caractère ":" n'est pas autorisé dans
//--- ouvre/crée la base de données dans le dossier commun du terminal
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
 Print("Base de données : échec de l'ouverture de ", filename, " avec le code ", GetLa
      return;
     }
   else
 Print("Base de données :", filename, " ouvert avec succès");

//--- crée la table TICKS
   if(!DatabaseExecute(db, "CREATE TABLE TICKS("
                        "SYMBOL              CHAR(10),"
                        "TIME                INT NOT NULL,"
                        "BID                 REAL,"
                        "ASK                 REAL,"
                        "LAST                REAL,"
                        "VOLUME              INT,"
                        "TIME_MSC            INT,"
```

© 2000-2024, MetaQuotes Ltd.

```
                        "VOLUME_REAL        REAL);"))
     {
 Print("DB :", filename, "échec de 'create table TICKS' avec le code ", GetLastError()
      DatabaseClose(db);
      return;
     }
//--- affiche la liste de tous les champs de la table TICKS
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(TICKS)", 0)<0)
     {
 PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(TICKS)\") a échoué, code d'erreur=%d à
      DatabaseClose(db);
      return;
     }
//--- crée une requête paramétrée pour ajouter des ticks à la table TICKS
   string sql="INSERT INTO TICKS (SYMBOL,TIME,BID,ASK,LAST,VOLUME,TIME_MSC,VOLUME_REAL
 " VALUES (?1,?2,?3,?4,?5,?6,?7,?8)"; // paramètres de la requête
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
 PrintFormat("DatabasePrepare() a échoué avec le code=%d", GetLastError());
      Print("Requête SQL : ", sql);
      DatabaseClose(db);
      return;
     }
//--- définit la valeur du premier paramètre de la requête
   DatabaseBind(request, 0, _Symbol);
//--- mémorise l'heure de début avant d'ajouter des ticks à la table TICKS
   start=GetTickCount();
   DatabaseTransactionBegin(db);
   int total=ArraySize(ticks);
   bool request_error=false;
   for(int i=0; i<total; i++)
     {
 //--- définit les valeurs des paramètres restants avant d'ajouter l'entrée
      ResetLastError();
      if(!DatabaseBind(request, 1, ticks[i].time))
        {
 PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
 PrintFormat("Tick #%d ligne=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
 //--- si l'appel précédent à DatabaseBind() a réussi, définit le paramètre suivant
      if(!request_error && !DatabaseBind(request, 2, ticks[i].bid))
        {
 PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
 PrintFormat("Tick #%d ligne=%d", i+1, __LINE__);
         request_error=true;
         break;
```

© 2000-2024, MetaQuotes Ltd.

```
        }
     if(!request_error && !DatabaseBind(request, 3, ticks[i].ask))
        {
PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
PrintFormat("Tick #%d ligne=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
     if(!request_error && !DatabaseBind(request, 4, ticks[i].last))
        {
PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
PrintFormat("Tick #%d ligne=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
     if(!request_error && !DatabaseBind(request, 5, ticks[i].volume))
        {
PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
PrintFormat("Tick #%d ligne=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
     if(!request_error && !DatabaseBind(request, 6, ticks[i].time_msc))
        {
PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
PrintFormat("Tick #%d ligne=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
     if(!request_error && !DatabaseBind(request, 7, ticks[i].volume_real))
        {
PrintFormat("DatabaseBind() a échoué avec le code=%d", GetLastError());
PrintFormat("Tick #%d ligne=%d", i+1, __LINE__);
         request_error=true;
         break;
        }

//--- exécute une requête d'insertion de l'entrée et recherche l'erreur
     if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_
        {
PrintFormat("DatabaseRead() a échoué avec le code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
//--- réinitialise la requête avant la prochaine mise à jour des paramètres
     if(!request_error && !DatabaseReset(request))
        {
PrintFormat("DatabaseReset() a échoué avec le code=%d", GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

```
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
 } //--- tous les ticks ont été parcourus

//--- statuts des transactions
   if(request_error)
     {
 PrintFormat("Table TICKS : échec de l'ajout de %d ticks", ArraySize(ticks));
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
 PrintFormat("Table TICKS : ajout de %d ticks en %d ms",
                  ArraySize(ticks), GetTickCount()-start);
     }

//--- ferme le fichier de base de données et information
   DatabaseClose(db);
 PrintFormat("Base de données : %s créée et fermée", filename);
  }
/*
 Résultat :
 EURUSD : CopyTicksRange a reçu 268061 ticks en 47 ms (de 2020.03.18 12:40 à 2020.03.1
 Base de données : EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite ouvert avec succè
  #| cid name        type     notnull dflt_value pk
  -+-----------------------------------------------
  1|   0 SYMBOL      CHAR(10)       0             0
  2|   1 TIME        INT            1             0
  3|   2 BID         REAL           0             0
  4|   3 ASK         REAL           0             0
  5|   4 LAST        REAL           0             0
  6|   5 VOLUME      INT            0             0
  7|   6 TIME_MSC    INT            0             0
  8|   7 VOLUME_REAL REAL           0             0
 Table TICKS : 268061 ticks ajoutés en 797 ms
 Base de données : EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite créée et fermée
  OnCalculateCorrelation=0.87 2020.03.19 13:00:  EURUSD vs GBPUSD  PERIOD_M30
*/
```

Voir aussi

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBindArray

© 2000-2024, MetaQuotes Ltd.

DatabaseBindArray

Définit un tableau comme valeur de paramètre.

```
bool  DatabaseBind(
   int  request,       // le handle d'une requête créée avec DatabasePrepare
   int  index,         // l'indice du paramètre dans la requête
   T&   array[]        // valeur de paramètre sous forme de tableau
   );
```

Paramètres

request

[in] Le handle d'une requête créée avec DatabasePrepare().

index

[in] L'indice du paramètre dans la requête pour lequel la valeur doit être définie. La numérotation commence à zéro.

array[]

[in] Le tableau à utiliser comme valeur du paramètre de la requête.

Valeur de Retour

Retourne true en cas de succès, sinon - false. Pour obtenir le code de l'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INVALID_PARAMETER (4003) - type non supporté ;
- ERR_ARRAY_BAD_SIZE (4011) - la taille du tableau en octets dépasse INT_MAX ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle de base de données invalide ;
- ERR_DATABASE_NOT_READY (5128) - ne peut pas utiliser la fonction pour faire une requête pour le moment (la requête est en cours d'exécution ou déjà terminée, DatabaseReset doit être appelé).

Note

La fonction est utilisée dans le cas où une requête SQL contient "?" ou "?N" valeurs paramétrables où N signifie l'indice du paramètre (à partir de un). Dans le même temps, l'indexation des paramètres dans DatabaseBindArray() commence à zéro.

Par exemple :

```
INSERT INTO table VALUES (?,?,?)
```

La fonction peut être appelée immédiatement après la création d'une demande paramétrée dans DatabasePrepare() ou après la réinitialisation de la demande à l'aide de DatabaseReset () .

Utilisez cette fonction avec DatabaseReset() pour exécuter la requête autant de fois que nécessaire avec différentes valeurs de paramètre.

Exemple :

```
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- ouvre la boîte de dialogue de sélection des fichiers avec l'extension DAT
   string selected_files[];
   if(!FileSelectDialog("Sélectionner les fichiers à télécharger", NULL,
                        "Fichiers Data (*.dat)|*.dat|Tous les fichiers (*.*)|*.*",
                        FSD_ALLOW_MULTISELECT, selected_files, "tester.dat")>0)
     {
      Print("Aucun fichier sélectionné. Sortie");
      return;
     }
//--- récupère la taille de tous les fichiers
   ulong filesize[];
   int filehandle[];
   int files=ArraySize(selected_files);
   ArrayResize(filesize, files);
   ZeroMemory(filesize);
   ArrayResize(filehandle, files);
   double total_size=0;
   for(int i=0; i<files; i++)
     {
      filehandle[i]=FileOpen(selected_files[i], FILE_READ|FILE_BIN);
      if(filehandle[i]!=INVALID_HANDLE)
        {
         filesize[i]=FileSize(filehandle[i]);
         //PrintFormat("%d, %s handle=%d %d bytes", i, selected_files[i], filehandle[i
         total_size+=(double)filesize[i];
        }
     }
//--- vérifie la taille globale des fichiers
   if(total_size==0)
     {
      PrintFormat("La taille totale des fichiers est 0. Sortie");
      return;
     }

//--- crée ou ouvre la base de données dans le dossier commun du terminal
   string filename="dat_files.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
 Print("DB: échec de l'ouverture de ", filename,  avec le code ", GetLastError());
      return;
     }
   else
 Print("Base de données :", filename, " ouvert avec succès");
//--- si la table FILES existe, la supprime
   if(DatabaseTableExists(db, "FILES"))
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- supprime la table
      if(!DatabaseExecute(db, "DROP TABLE FILES"))
        {
 Print("Impossible de supprimer la table FILES avec le code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- crée la table FILES
   if(!DatabaseExecute(db, "CREATE TABLE FILES("
                        "NAME           TEXT NOT NULL,"
                        "SIZE           INT  NOT NULL,"
                        "PERCENT_SIZE   REAL NOT NULL,"
                        "DATA           BLOB NOT NULL);"))
     {
 Print("DB: impossible de créer la table FILES avec le code",  GetLastError());
      DatabaseClose(db);
      return;
     }
//--- affiche la liste de tous les champs de la table FILES
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(FILES)", 0)<0)
     {
 PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(FILES)\") a échoué, code d'erreur=%d à
      DatabaseClose(db);
      return;
     }

//--- créer une requête paramétrée pour ajouter des fichiers dans la table FILES
   string sql="INSERT INTO FILES (NAME,SIZE,PERCENT_SIZE,DATA)"
              " VALUES (?1,?2,?3,?4);"; // paramètres de la requête
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
 PrintFormat("DatabasePrepare() a échoué avec le code=%d", GetLastError());
      Print("Requête SQL : ", sql);
      DatabaseClose(db);
      return;
     }

//--- parcourt tous les fichiers et les ajoute à la table FILES
   bool request_error=false;
   DatabaseTransactionBegin(db);
   int count=0;
   uint size;
   for(int i=0; i<files; i++)
     {
      if(filehandle[i]!=INVALID_HANDLE)
        {
         char data[];
```

```
        size=FileReadArray(filehandle[i], data);
        if(size==0)
          {
           PrintFormat("FileReadArray(%s) a échoué avec le code %d", selected_files[i
           continue;
          }

        count++;
        //--- définit les valeurs des paramètres avant d'ajouter le fichier à la tabl
        if(!DatabaseBind(request, 0, selected_files[i]))
          {
           PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LIN
           request_error=true;
           break;
          }
        if(!DatabaseBind(request, 1, size))
          {
           PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LIN
           request_error=true;
           break;
          }
        if(!DatabaseBind(request, 2, double(size)*100./total_size))
          {
           PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LIN
           request_error=true;
           break;
          }
        if(!DatabaseBindArray(request, 3, data))
          {
           PrintFormat("DatabaseBind() a échoué à la ligne %d avec le code=%d", __LIN
           request_error=true;
           break;
          }
//--- exécute une requête d'insertion de l'entrée et recherche l'erreur
        if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
          {
           PrintFormat("DatabaseRead() a échoué avec le code=%d", GetLastError());
           DatabaseFinalize(request);
           request_error=true;
           break;
          }
        else
           PrintFormat("%d. %s: %d bytes", count, selected_files[i],size);
        //--- réinitialise la requpete avant la prochaine mise à jour des paramètres
        if(!DatabaseReset(request))
          {
           PrintFormat("DatabaseReset() a échoué avec le code=%d", GetLastError());
           DatabaseFinalize(request);
           request_error=true;
```

© 2000-2024, MetaQuotes Ltd.

```
            break;
           }
        }
     }
//--- statuts des transactions
   if(request_error)
     {
      PrintFormat("Table FILES : échec de l'ajout de %d fichiers", count);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table FILES : %d fichiers ajoutés", count);
     }

//--- ferme le fichier de base de données et information
   DatabaseClose(db);
 PrintFormat("Base de données : %s créée et fermée", filename);
  }
```

Voir aussi

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBind

© 2000-2024, MetaQuotes Ltd.

DatabaseRead

Se déplace sur l'enregistrement suivant des résultats de la requête.

```
bool  DatabaseRead(
   int  request      // handle de la requête retourné par DatabasePrepare
   );
```

Paramètres

request

[in] Handle de la requête retourné par DatabasePrepare().

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INVALID_PARAMETER (4003) - aucun nom de table spécifié (chaîne vide ou NULL) ;
- ERR_WRONG_STRING_PARAMETER (5040) - erreur de conversion de la requête en UTF-8 ;
- ERR_DATABASE_INTERNAL (5120) - erreur interne de la base de données ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données ;
- ERR_DATABASE_EXECUTE (5124) - erreur d'exécution de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) - aucune table n'existe (ceci n'est pas une erreur, mais une fin normale).

Voir aussi

DatabasePrepare, DatabaseReadBind

© 2000-2024, MetaQuotes Ltd.

DatabaseReadBind

Se déplace sur l'enregistrement suivant et écrit les données dans la structure passée.

```
bool  DatabaseReadBind(
   int    request,           // le handle d'une requête créée avec DatabasePrepare
   void&  struct_object      // référence sur une structure pour récupérer les données
   );
```

Paramètres

request

[in] Le handle d'une requête créée avec DatabasePrepare().

struct_object

[out] La référence sur une structure pour récupérer les données. La structure ne doit contenir que des types numériques et/ou chaînes de caractères (les tableaux ne sont pas autorisés) comme membres et ne peuvent pas être des descendants.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INVALID_PARAMETER (4003) - aucun nom de table spécifié (chaîne vide ou NULL) ;
- ERR_WRONG_STRING_PARAMETER (5040) - erreur de conversion de la requête en UTF-8 ;
- ERR_DATABASE_INTERNAL (5120) - erreur interne de la base de données ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données ;
- ERR_DATABASE_EXECUTE (5124) - erreur d'exécution de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) - aucune table n'existe (ceci n'est pas une erreur, mais une fin normale).

Note

Le nombre de colonnes dans la structure *struct_object* ne doit pas être supérieur à DatabaseColumnsCount(). Si le nombre de colonnes de la structure *struct_object* est inférieur au nombre de champs dans l'enregistrement, une lecture partielle est effectuée. Les données restantes peuvent être récupérées explicitement en utilisant les fonctions correspondantes DatabaseColumnText(), DatabaseColumnInteger() et autres.

Exemple:

```
struct Person
  {
   int               id;
   string            name;
   int               age;
   string            address;
   double            salary;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   int db;
   string filename="company.sqlite";
//--- open
   db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATABASE_
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- if the table COMPANY exists then drop the table
   if(DatabaseTableExists(db, "COMPANY"))
     {
      //--- delete the table
      if(!DatabaseExecute(db, "DROP TABLE COMPANY"))
        {
         Print("Failed to drop table COMPANY with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- create table
   if(!DatabaseExecute(db, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
     {
      Print("DB: ", filename, " create table failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- insert data
   if(!DatabaseExecute(db, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'A
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'T
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'M
     {
      Print("DB: ", filename, " insert failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- prepare the request
   int request=DatabasePrepare(db, "SELECT * FROM COMPANY WHERE SALARY>15000");
```

© 2000-2024, MetaQuotes Ltd.

```
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- print records
   Person person;
   Print("Persons with salary > 15000:");
   for(int i=0; DatabaseReadBind(request, person); i++)
      Print(i, ":  ", person.id, " ", person.name, " ", person.age, " ", person.addres
//--- delete request after use
   DatabaseFinalize(request);

   Print("Some statistics:");
//--- prepare new request about total salary
   request=DatabasePrepare(db, "SELECT SUM(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double total_salary;
      DatabaseColumnDouble(request, 0, total_salary);
      Print("Total salary=", total_salary);
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- prepare new request about average salary
   request=DatabasePrepare(db, "SELECT AVG(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      ResetLastError();
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double aver_salary;
      DatabaseColumnDouble(request, 0, aver_salary);
      Print("Average salary=", aver_salary);
     }
//--- delete request after use
   DatabaseFinalize(request);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- close database
   DatabaseClose(db);
  }
//+------------------------------------------------------------------
/*
Output:
Persons with salary > 15000:
0:  1 Paul 32 California 25000.0
1:  3 Teddy 23 Norway 20000.0
2:  4 Mark 25 Rich-Mond  65000.0
Some statistics:
Total salary=125000.0
Average salary=31250.0
*/
```

Voir aussi

DatabasePrepare, DatabaseRead

© 2000-2024, MetaQuotes Ltd.

DatabaseFinalize

Supprime une requête créée avec DatabasePrepare().

```
void  DatabaseFinalize(
   int  request      // handle de la requête retourné par DatabasePrepare
   );
```

Paramètres

database

[in] Handle de la requête retourné par DatabasePrepare().

Valeur de Retour

Aucun.

Note

Si le handle est invalide, la fonction retourne l'erreur ERR_DATABASE_INVALID_HANDLE. Vous pouvez vérifier l'erreur en appelant GetLastError().

Voir aussi

DatabasePrepare, DatabaseExecute

© 2000-2024, MetaQuotes Ltd.

DatabaseTransactionBegin

Démarre l'exécution d'une transaction.

```
bool  DatabaseTransactionBegin(
   int  database      // handle de la base de données retourné par DatabaseOpen
   );
```

Paramètres

database

[in] Handle de la base de données retourné par DatabaseOpen().

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INTERNAL_ERROR (4001) - erreur critique d'exécution ;
- ERR_INVALID_PARAMETER (4003) - le paramètre sql contient une chaîne vide ;
- ERR_NOT_ENOUGH_MEMORY (4004) - mémoire insuffisante ;
- ERR_WRONG_STRING_PARAMETER (5040) - erreur de conversion de la requête en UTF-8 ;
- ERR_DATABASE_INTERNAL (5120) - erreur interne de la base de données ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données ;
- ERR_DATABASE_EXECUTE (5124) - erreur d'exécution de la requête.

Note

La fonction DatabaseTransactionBegin() doit être appelée avant l'exécution d'une transaction. Toute transaction doit être démarrée avec DatabaseTransactionBegin() et terminée avec DatabaseTransactionCommit().

Example:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=AccountInfoString(ACCOUNT_SERVER) +"_"+IntegerToString(AccountInfo
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("Deals in the trading history: %d ", deals_total);

//--- measure the transaction execution speed using DatabaseTransactionBegin/Database
   ulong start=GetMicrosecondCount();
   bool fast_transactions=true;
   InsertDeals(db, fast_transactions);
   double fast_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit

//--- delete the DEALS table, and then create it again
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
//--- create a new DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }

//--- test again, this time without using DatabaseTransactionBegin/DatabaseTransactio
   fast_transactions=false;
   start=GetMicrosecondCount();
   InsertDeals(db, fast_transactions);
   double slow_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit
//--- report gain in time
   PrintFormat("Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided acc
//--- close the database
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2737
   Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit: time=48.5 r
```

```
   Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit: time=25818.
   Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided acceleration by
*/
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop table DEALS with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
        {
         Print("DB: create table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
```

```
bool InsertDeals(int database, bool begintransaction=true)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was exec
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number
   long     reason;              // deal execution reason or source
//--- go through all deals and add to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
//--- if fast transaction performance method is used
   if(begintransaction)
     {
      // --- lock the database before executing transactions
      DatabaseTransactionBegin(database);
     }
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- add each deal using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                       deal_ticket, order_ticket, position_ticket, tim
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%dwith code %d", __FUNCTION__, deal_t
```

```
          PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
          failed=true;
          break;
         }
      }
//--- check for transaction execution errors
   if(failed)
     {
      //--- if fast transaction performance method is used
      if(begintransaction)
        {
         //--- roll back all transactions and unlock the database
         DatabaseTransactionRollback(database);
        }
      Print("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastError());
      return(false);
     }
//--- if fast transaction performance method is used
   if(begintransaction)
     {
      //--- all transactions have been performed successfully - record changes and unl
      DatabaseTransactionCommit(database);
     }
//--- successful completion
   return(true);
  }
//+------------------------------------------------------------------+
```

Voir aussi

DatabaseExecute, DatabasePrepare, DatabaseTransactionCommit, DatabaseTransactionRollback

© 2000-2024, MetaQuotes Ltd.

DatabaseTransactionCommit

Termine l'exécution d'une transaction.

```
bool  DatabaseTransactionCommit(
   int  database      // handle de la base de données retourné par DatabaseOpen
   );
```

Paramètres

database

[in] Handle de la base de données retourné par DatabaseOpen().

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INTERNAL_ERROR (4001) - erreur critique d'exécution ;
- ERR_INVALID_PARAMETER (4003) - le paramètre sql contient une chaîne vide ;
- ERR_NOT_ENOUGH_MEMORY (4004) - mémoire insuffisante ;
- ERR_WRONG_STRING_PARAMETER (5040) - erreur de conversion de la requête en UTF-8 ;
- ERR_DATABASE_INTERNAL (5120) - erreur interne de la base de données ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données ;
- ERR_DATABASE_EXECUTE (5124) - erreur d'exécution de la requête.

Note

La fonction DatabaseTransactionCommit() compléte toutes les transactions exécutées après l'appel à la fonction DatabaseBeginTransaction(). Toute transaction doit être démarrée avec DatabaseTransactionBegin() et terminée avec DatabaseTransactionCommit() pour une fin réussie.

Voir aussi

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionRollback

© 2000-2024, MetaQuotes Ltd.

DatabaseTransactionRollback

Annule une transaction.

```
bool  DatabaseTransactionRollback(
   int  database      // handle de la base de données retourné par DatabaseOpen
   );
```

Paramètres

database

[in] Handle de la base de données retourné par DatabaseOpen().

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_INTERNAL_ERROR (4001) - erreur critique d'exécution ;
- ERR_INVALID_PARAMETER (4003) - le paramètre sql contient une chaîne vide ;
- ERR_NOT_ENOUGH_MEMORY (4004) - mémoire insuffisante ;
- ERR_WRONG_STRING_PARAMETER (5040) - erreur de conversion de la requête en UTF-8 ;
- ERR_DATABASE_INTERNAL (5120) - erreur interne de la base de données ;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle invalide de la base de données ;
- ERR_DATABASE_EXECUTE (5124) - erreur d'exécution de la requête.

Note

L'appel à DatabaseTransactionRollback() annule toutes les transactions exécutées après l'appel à la fonction DatabaseTransactionBegin(). La fonction DatabaseTransactionRollback() est nécessaire pour annuler tous les changements dans une base de données en cas d'erreurs lors de l'exécution d'une transaction.

Voir aussi

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionCommit

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnsCount

Retourne le nombre de colonnes dans une requête.

```
int  DatabaseColumnsCount(
   int  request      // handle de la requête retourné par DatabasePrepare
   );
```

Paramètres

request

[in] Handle de la requête retourné par DatabasePrepare().

Valeur de Retour

Le nombre de colonnes ou -1 en cas d'erreur. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_DATABASE_INVALID_HANDLE (5121) – handle invalide de la requête.

Note

Il n'est pas utile d'appeler la fonction DatabaseRead() pour obtenir le nombre de colonnes d'une requête créée avec DatabasePrepare(). Pour les autres fonctions DatabaseColumnXXX(), DatabaseRead() doit être appelée au préalable.

Voir aussi

DatabasePrepare, DatabaseFinalize, DatabaseClose

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnName

Retourne le nom d'une colonne par son numéro.

```
bool  DatabaseColumnName(
   int      request,     // handle de la requête retourné par DatabasePrepare
   int      column,      // indice de la colonne dans la requête
   string&  name         // référence sur la variable pour récupérer la valeur
   );
```

Paramètres

request

[in] Handle de la requête retourné par DatabasePrepare().

column

[in] Indice de la colonne dans la requête. La numérotation des champs démarre à 0 et ne peut pas être supérieure à DatabaseColumnsCount() - 1.

name

[out] Variable pour écrire le nom de la colonne.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_DATABASE_INVALID_HANDLE (5121) – handle invalide de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) - l'indice de 'colonne' est supérieur à DatabaseColumnsCount() -1.

Note

La valeur ne peut être obtenue que si au moins un appel à DatabaseRead() a été fait auparavant pour 'request'.

Voir aussi

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnType

Retourne le type d'une colonne par son numéro.

```
ENUM_DATABASE_FIELD_TYPE  DatabaseColumnType(
   int  request,      // handle de la requête retourné par DatabasePrepare
   int  column        // indice de la colonne dans la requête
   );
```

Paramètres

request

[in] Handle de la requête retourné par DatabasePrepare().

column

[in] Indice de la colonne dans la requête. La numérotation des champs démarre à 0 et ne peut pas être supérieure à DatabaseColumnsCount() - 1.

Valeur de Retour

Retourne le type de la colonne dans l'énumération ENUM_DATABASE_FIELD_TYPE. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_DATABASE_INVALID_HANDLE (5121) – handle invalide de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) – l'indice de 'colunne' est supérieur à DatabaseColumnsCount() -1.

Note

La valeur ne peut être obtenue que si au moins un appel à DatabaseRead() a été fait auparavant pour 'request'.

ENUM_DATABASE_FIELD_TYPE

ID	Description
DATABASE_FIELD_TYPE_INVALID	Erreur lors de l'obtention du type, le code d'erreur peut être obtenu en utilisant int GetLastError()
DATABASE_FIELD_TYPE_INTEGER	Type integer
DATABASE_FIELD_TYPE_FLOAT	Type réel
DATABASE_FIELD_TYPE_TEXT	Type string
DATABASE_FIELD_TYPE_BLOB	Type binaire
DATABASE_FIELD_TYPE_NULL	Type spécial NULL

Voir aussi

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnSize

Retourne la taille d'une colonne en octets.

```
int  DatabaseColumnSize(
   int  request,     // handle de la requête retourné par DatabasePrepare
   int  column       // indice de la colonne dans la requête
   );
```

Paramètres

request

[in] Handle de la requête retourné par DatabasePrepare().

column

[in] Indice de la colonne dans la requête. La numérotation des champs démarre à 0 et ne peut pas être supérieure à DatabaseColumnsCount() - 1.

Valeur de Retour

La taille de la colonne en octets est retounée en cas de succès, sinon -1. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_DATABASE_INVALID_HANDLE (5121) – handle invalide de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) – l'indice de 'colunne' est supérieur à DatabaseColumnsCount() -1.

Note

La valeur ne peut être obtenue que si au moins un appel à DatabaseRead() a été fait auparavant pour 'request'.

Voir aussi

DatabasePrepare, DatabaseColumnBlob, DatabaseColumnsCount, DatabaseColumnName, DatabaseColumnType

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnText

Retourne la valeur d'une colonne de type string de l'enregistrement courant.

```
bool  DatabaseColumnText(
   int      request,     // handle de la requête retourné par DatabasePrepare
   int      column,      // indice de la colonne dans la requête
   string&  value        // référence sur la variable pour récupérer la valeur
   );
```

Paramètres

request

[in] Handle de la requête retourné par DatabasePrepare().

column

[in] Indice de la colonne dans la requête. La numérotation des champs démarre à 0 et ne peut pas être supérieure à DatabaseColumnsCount() - 1.

value

[out] Référence sur la variable pour récupérer la valeur.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_DATABASE_INVALID_HANDLE (5121) – handle invalide de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) - l'indice de 'colunne' est supérieur à DatabaseColumnsCount() -1.

Note

La valeur ne peut être obtenue que si au moins un appel à DatabaseRead() a été fait auparavant pour 'request'.

Pour lire la valeur de l'enregistrement suivant, appelez d'abord DatabaseRead().

Voir aussi

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnInteger

Retourne la valeur d'une colonne de type int de l'enregistrement courant.

```
bool  DatabaseColumnInteger(
   int   request,      // handle de la requête retourné par DatabasePrepare
   int   column,       // indice de la colonne dans la requête
   int&  value         // référence sur la variable pour récupérer la valeur
   );
```

Paramètres

request

[in] Handle de la requête retourné par DatabasePrepare().

column

[in] Indice de la colonne dans la requête. La numérotation des champs démarre à 0 et ne peut pas être supérieure à DatabaseColumnsCount() - 1.

value

[out] Référence sur la variable pour récupérer la valeur.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_DATABASE_INVALID_HANDLE (5121) – handle invalide de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) - l'indice de 'colonne' est supérieur à DatabaseColumnsCount() -1.

Note

La valeur ne peut être obtenue que si au moins un appel à DatabaseRead() a été fait auparavant pour 'request'.

Pour lire la valeur de l'enregistrement suivant, appelez d'abord DatabaseRead().

Voir aussi

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnLong

Retourne la valeur d'une colonne de type long de l'enregistrement courant.

```
bool  DatabaseColumnLong(
   int    request,     // handle de la requête retourné par DatabasePrepare
   int    column,      // indice de la colonne dans la requête
   long&  value        // référence sur la variable pour récupérer la valeur
   );
```

Paramètres

request

[in] Handle de la requête retourné par DatabasePrepare().

column

[in] Indice de la colonne dans la requête. La numérotation des champs démarre à 0 et ne peut pas être supérieure à DatabaseColumnsCount() - 1.

value

[out] Référence sur la variable pour récupérer la valeur.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_DATABASE_INVALID_HANDLE (5121) – handle invalide de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) - l'indice de 'colunne' est supérieur à DatabaseColumnsCount() -1.

Note

La valeur ne peut être obtenue que si au moins un appel à DatabaseRead() a été fait auparavant pour 'request'.

Pour lire la valeur de l'enregistrement suivant, appelez d'abord DatabaseRead().

Voir aussi

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnDouble

Retourne la valeur d'une colonne de type double de l'enregistrement courant.

```
bool  DatabaseColumnDouble(
   int      request,     // handle de la requête retourné par DatabasePrepare
   int      column,      // indice de la colonne dans la requête
   double&  value        // une référence sur la variable pour récupérer la valeur
   );
```

Paramètres

request

[in] Handle de la requête retourné par DatabasePrepare().

column

[in] Indice de la colonne dans la requête. La numérotation des champs démarre à 0 et ne peut pas être supérieure à DatabaseColumnsCount() - 1.

value

[out] Référence sur la variable pour récupérer la valeur.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_DATABASE_INVALID_HANDLE (5121) – handle invalide de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) - l'indice de 'colonne' est supérieur à DatabaseColumnsCount() -1.

Note

La valeur ne peut être obtenue que si au moins un appel à DatabaseRead() a été fait auparavant pour 'request'.

Pour lire la valeur de l'enregistrement suivant, appelez d'abord DatabaseRead().

Voir aussi

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnBlob

Retourne le tableau de caractères d'une colonne de type BLOB de l'enregistrement courant.

```
bool  DatabaseColumnBlob(
   int    request,      // handle de la requête retourné par DatabasePrepare
   int    column,       // indice de la colonne dans la requête
   void&  data[]        // une référence sur la variable pour récupérer la valeur
   );
```

Paramètres

request

[in] Handle de la requête retourné par DatabasePrepare().

column

[in] Indice de la colonne dans la requête. La numérotation des champs démarre à 0 et ne peut pas être supérieure à DatabaseColumnsCount() - 1.

data[]

[out] Référence sur le tableau pour récupérer la valeur.

Valeur de Retour

Retourne true en cas de succès, sinon false. Pour obtenir le code d'erreur, utilisez GetLastError(), les réponses possibles sont :

- ERR_DATABASE_INVALID_HANDLE (5121) – handle invalide de la requête ;
- ERR_DATABASE_NO_MORE_DATA (5126) - l'indice de 'colonne' est supérieur à DatabaseColumnsCount() -1.

Note

La valeur ne peut être obtenue que si au moins un appel à DatabaseRead() a été fait auparavant pour 'request'.

Pour lire la valeur de l'enregistrement suivant, appelez d'abord DatabaseRead().

Voir aussi

DatabasePrepare, DatabaseColumnSize, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

Utilisation de DirectX

Les fonctions et shaders DirectX 11 sont conçus pour la visualisation 3D directement sur un graphique des prix.

La création de graphiques 3D nécessite un contexte graphique (DXContextCreate) avec la taille d'image correspondante. En outre, il est nécessaire de préparer les buffers de vertex et d'index (DXBufferCreate), ainsi que de créer les sommets et les shaders de pixel (DXShaderCreate). C'est tout ce qu'il suffit pour afficher des graphiques en couleur.

Le niveau graphique suivant nécessite les entrées (DXInputSet) pour passer des paramètres de rendus supplémentaires aux shaders. Ceci permet de définir le placement de la caméra et des objets 3D, de décrire les sources de lumière et d'implémenter le contrôle de la souris et du clavier.

Les fonctions MQL5 intégrées vous permettent donc de créer des graphiques 3D animés directement dans MetaTrader 5 sans devoir utiliser des outils externes. Votre carte vidéo doit supporter DX 11 et Shader Model 5.0 pour que les fonctions puissent être utilisées.

Pour commencer à travailler avec la bibliothèque, lisez simplement l'article Comment créer des graphiques 3D à l'aide de DirectX dans MetaTrader 5.

Fonction	Action
DXContextCreate	Crée un contexte gaphique pour le rendu des cadres de la taille spécifiée
DXContextSetSize	Change la taille du cadre du contexte graphique créé avec DXContextCreate()
DXContextGetSize	Retourne la taille du cadre d'un contexte graphique créé avec DXContextCreate()
DXContextClearColors	Applique la couleur spécifiée à tous les pixels pour le buffer de rendu
DXContextClearDepth	Efface le buffer de profondeur
DXContextGetColors	Retourne l'image de la taille et du décalage spécifiés pour un contexte graphique
DXContextGetDepth	Retourne le buffer de profondeur du cadre rendu
DXBufferCreate	Crée le buffer d'un type spécifié basé sur un tableau de données
DXTextureCreate	Crée une texture 2D à partir d'un rectangle de la taille spécifiée à partir de l'image passée
DXInputCreate	Crée les entrées d'un shader
DXInputSet	Définit les entrées d'un shader
DXShaderCreate	Crée un shader du type spécifié
DXShaderSetLayout	Définit la mise en page du sommet pour le shader du sommet

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
DXShaderInputsSet	Définit les entrées d'un shader
DXShaderTexturesSet	Définit les textures du shader
DXDraw	Effectue le rendering des sommets du buffer du sommet définit avec DXBufferSet()
DXDrawIndexed	Effectue le rendering des primitives graphiques décrites par le buffer d'index depuis DXBufferSet()
DXPrimiveTopologySet	Définit le type des primitives du rendu en utilisant DXDrawIndexed()
DXBufferSet	Définit le buffer pour le rendu courant
DXShaderSet	Définit un shader pour le rendu
DXHandleType	Retourne le type du handle
DXRelease	Libère un handle

© 2000-2024, MetaQuotes Ltd.

DXContextCreate

Crée un contexte gaphique pour le rendu des cadres de la taille spécifiée.

```
int  DXContextCreate(
   uint  width,      // largeur en pixels
   uint  height      // hauteur en pixels
   );
```

Paramètres

width

[in] Largeur du cadre en pixels.

height

[in] Hauteur du cadre en pixels.

Valeur de Retour

Le handle sur le contexte créé ou INVALID_HANDLE en cas d'erreur. Pour obtenir le code d'erreur, utilisez la fonction GetLastError().

Note

Tous les objets graphiques créés avec les fonctions DXBufferCreate, DXInputCreate, DXShaderCreate et DXTextureCreate ne peuvent être utilisés qu'avec le contexte graphique dans lequel ils ont été créés.

La taille d'un cadre peut ensuite être changée avec DXContextSetSize().

Un handle créé qui n'est plus utilisé doit être libéré explicitement avec la fonction DXRelease().

© 2000-2024, MetaQuotes Ltd.

DXContextSetSize

Change la taille du cadre du contexte graphique créé avec DXContextCreate().

```
bool  DXContextSetSize(
   int    context,       // handle du contexte graphique
   uint&  width,         // largeur en pixels
   uint&  height         // hauteur en pixels
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

width

[in] Largeur du cadre en pixels.

height

[in] Hauteur du cadre en pixels.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

La taille du cadre d'un contexte graphique ne peut être changée qu'entre deux rendus du cadre.

© 2000-2024, MetaQuotes Ltd.

DXContextGetSize

Retourne la taille du cadre d'un contexte graphique créé avec DXContextCreate().

```
bool  DXContextGetSize(
   int    context,       // handle du contexte graphique
   uint&  width,         // largeur en pixels
   uint&  height         // hauteur en pixels
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

width

[out] Largeur du cadre en pixels.

height

[out] Hauteur du cadre en pixels.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

DXContextClearColors

Applique la couleur spécifiée à tous les pixels pour le buffer de rendu.

```
bool  DXContextClearColors(
   int              context,       // handle du contexte graphique
   const DXVector&  color          // couleur
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

color

[in] Couleur du rendu.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

La fonction DXContextClearColors() peut être utilisée pour effacer le buffer de couleur avant d'effectuer le rendu du cadre suivant.

© 2000-2024, MetaQuotes Ltd.

DXContextClearDepth

Efface le buffer de profondeur.

```
bool  DXContextClearDepth(
   int  context      // handle du contexte graphique
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

La fonction DXContextClearDepth() peut être utilisée pour effacer le buffer de profondeur avant d'effectuer le rendu du cadre suivant.

© 2000-2024, MetaQuotes Ltd.

DXContextGetColors

Retourne l'image de la taille et du décalage spécifiés pour un contexte graphique.

```
bool  DXContextGetColors(
   int    context,                      // handle du contexte graphique
   uint&  image[],                      // tableau de pixels de l'image
   int    image_width=WHOLE_ARRAY,      // largeur de l'image en pixels
   int    image_height=WHOLE_ARRAY,     // hauteur de l'image en pixels
   int    image_offset_x=0,             // décalage X
   int    image_offset_y=0              // décalage Y
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

image

[out] Le tableau de *image_width*image_height* pixels au format ARGB.

image_width=WHOLE_ARRAY

[in] Largeur de l'image en pixels.

image_height=WHOLE_ARRAY

[in] Hauteur de l'image en pixels.

image_offset_x=0

[in] Décalage X.

image_offset_y=0

[in] Décalage Y.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

DXContextGetDepth

Retourne le buffer de profondeur du cadre rendu.

```
bool  DXContextGetDepth(
   int     context,     // handle du contexte graphique
   float&  image[]      // tableau des valeurs de profondeur
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

image

[out] Tableau des valeurs du buffer de profondeur du cadre sur lequel le rendu a été réalisé.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

Le buffer retourné contient la profondeur de chaque pixel du cadre rendu, qui peut être obtenu avec DXContextGetColors() en unités relatives (de 0,0 à 1,0).

© 2000-2024, MetaQuotes Ltd.

DXBufferCreate

Crée le buffer d'un type spécifié basé sur un tableau de données.

```
int  DXBufferCreate(
   int                  context,              // handle du contexte graphique
   ENUM_DX_BUFFER_TYPE  buffer_type,          // type du buffer créé
   const void&          data[],               // données du buffer
   uint                 start=0,              // index initial
   uint                 count=WHOLE_ARRAY     // nombre d'éléments
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

buffer_type

[in] Type du buffer de l'énumération ENUM_DX_BUFFER_TYPE.

data[]

[in] Données pour créer le buffer.

start

[in] Indice du premier élément du tableau, à partir duquel les valeurs du tableau sont utilisées pour créer le buffer. Par défaut, les données sont prises depuis le début du tableau.

count

[in] Nombre de valeurs. Par défaut, le tableau est utilisé en entier (count=WHOLE_ARRAY).

Valeur de Retour

Le handle du buffer créé ou INVALID_HANDLE en cas d'erreur. Pour obtenir le code d'erreur, utilisez la fonction GetLastError().

Note

Pour le buffer d'index, le tableau *data[]* doit être de type 'uint', tandis que le buffer de sommets contient le tableau de structures décrivant les sommets.

Un handle créé qui n'est plus utilisé doit être libéré explicitement avec la fonction DXRelease().

ENUM_DX_BUFFER_TYPE

ID	Valeur	Description
DX_BUFFER_VERTEX	1	Buffer du sommet
DX_BUFFER_INDEX	2	Buffer des index

© 2000-2024, MetaQuotes Ltd.

DXTextureCreate

Crée une texture 2D à partir d'un rectangle de la taille spécifiée à partir de l'image passée.

```
int  DXTextureCreate(
   int             context,      // handle du contexte graphique
   ENUM_DX_FORMAT  format,       // format de la couleur du pixel
   uint            width,        // largeur de l'image source
   uint            height,       // hauteur de l'image source
   const void&     data[],       // tableau de pixels de l'image source
   uint            data_x,       // coordonnée X du rectangle utilisé pour créer une t
   uint            data_y,       // coordonnée Y du rectangle utilisé pour créer une t
   uint            data_width,   // largeur du rectangle pour créer une texture
   uint            data_height   // hauteur du rectangle pour créer une texture
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

format

[in] Format de couleur du pixel depuis l'énumération ENUM_DX_FORMAT.

width

[in] Largeur de l'image sur laquelle la texture est basée.

height

[in] Hauteur de l'image sur laquelle la texture est basée.

data

[in] Tableau de pixels de l'image sur laquelle la texture est basée.

data_x

[in] Coordonnée X d'un rectangle (décalage X) utilisé pour créer une texture.

data_y

[in] Coordonnée Y d'un rectangle (décalage Y) utilisé pour créer une texture.

data_width

[in] Largeur d'un rectangle utilisé pour créer une texture.

data_height

[in] Hauteur d'un rectangle utilisé pour créer une texture.

Valeur de Retour

Handle de la texture ou INVALID_HANDLE en cas d'erreur. Pour obtenir le code d'erreur, utilisez la fonction GetLastError().

Note

Un handle créé qui n'est plus utilisé doit être libéré explicitement avec la fonction DXRelease().

© 2000-2024, MetaQuotes Ltd.

ENUM_DX_FORMAT

ID	Valeur	Correspondant au format DXGI_FORMAT
DX_FORMAT_UNKNOWN	0	DXGI_FORMAT_UNKNOWN
DX_FORMAT_R32G32B32A32_TYPELESS	1	DXGI_FORMAT_R32G32B32A32_TYPELESS
DX_FORMAT_R32G32B32A32_FLOAT	2	DXGI_FORMAT_R32G32B32A32_FLOAT
DX_FORMAT_R32G32B32A32_UINT	3	DXGI_FORMAT_R32G32B32A32_UINT
DX_FORMAT_R32G32B32A32_SINT	4	DXGI_FORMAT_R32G32B32A32_SINT
DX_FORMAT_R32G32B32_TYPELESS	5	DXGI_FORMAT_R32G32B32_TYPELESS
DX_FORMAT_R32G32B32_FLOAT	6	DXGI_FORMAT_R32G32B32_FLOAT
DX_FORMAT_R32G32B32_UINT	7	DXGI_FORMAT_R32G32B32_UINT
DX_FORMAT_R32G32B32_SINT	8	DXGI_FORMAT_R32G32B32_SINT
DX_FORMAT_R16G16B16A16_TYPELESS	9	DXGI_FORMAT_R16G16B16A16_TYPELESS
DX_FORMAT_R16G16B16A16_FLOAT	10	DXGI_FORMAT_R16G16B16A16_FLOAT
DX_FORMAT_R16G16B16A16_UNORM	11	DXGI_FORMAT_R16G16B16A16_UNORM
DX_FORMAT_R16G16B16A16_UINT	12	DXGI_FORMAT_R16G16B16A16_UINT
DX_FORMAT_R16G16B16A16_SNORM	13	DXGI_FORMAT_R16G16B16A16_SNORM
DX_FORMAT_R16G16B16A16_SINT	14	DXGI_FORMAT_R16G16B16A16_SINT
DX_FORMAT_R32G32_TYPELESS	15	DXGI_FORMAT_R32G32_TYPELESS
DX_FORMAT_R32G32_FLOAT	16	DXGI_FORMAT_R32G32_FLOAT
DX_FORMAT_R32G32_UINT	17	DXGI_FORMAT_R32G32_UINT
DX_FORMAT_R32G32_SINT	18	DXGI_FORMAT_R32G32_SINT
DX_FORMAT_R32G8X24_TYPELESS	19	DXGI_FORMAT_R32G8X24_TYPELESS
DX_FORMAT_D32_FLOAT_S8X24_UINT	20	DXGI_FORMAT_D32_FLOAT_S8X24_UINT
DX_FORMAT_R32_FLOAT_X8X24_TYPELESS	21	DXGI_FORMAT_R32_FLOAT_X8X24_TYPELESS
DX_FORMAT_X32_TYPELESS_G8X24_UINT	22	DXGI_FORMAT_X32_TYPELESS_G8X24_UINT
DX_FORMAT_R10G10B10A2_TYPELESS	23	DXGI_FORMAT_R10G10B10A2_TYPELESS

© 2000-2024, MetaQuotes Ltd.

ID	Valeur	Correspondant au format DXGI_FORMAT
DX_FORMAT_R10G10B10A2_UNORM	24	DXGI_FORMAT_R10G10B10A2_UNORM
DX_FORMAT_R10G10B10A2_UINT	25	DXGI_FORMAT_R10G10B10A2_UINT
DX_FORMAT_R11G11B10_FLOAT	26	DXGI_FORMAT_R11G11B10_FLOAT
DX_FORMAT_R8G8B8A8_TYPELESS	27	DXGI_FORMAT_R8G8B8A8_TYPELESS
DX_FORMAT_R8G8B8A8_UNORM	28	DXGI_FORMAT_R8G8B8A8_UNORM
DX_FORMAT_R8G8B8A8_UNORM_SRGB	29	DXGI_FORMAT_R8G8B8A8_UNORM_SRGB
DX_FORMAT_R8G8B8A8_UINT	30	DXGI_FORMAT_R8G8B8A8_UINT
DX_FORMAT_R8G8B8A8_SNORM	31	DXGI_FORMAT_R8G8B8A8_SNORM
DX_FORMAT_R8G8B8A8_SINT	32	DXGI_FORMAT_R8G8B8A8_SINT
DX_FORMAT_R16G16_TYPELESS	33	DXGI_FORMAT_R16G16_TYPELESS
DX_FORMAT_R16G16_FLOAT	34	DXGI_FORMAT_R16G16_FLOAT
DX_FORMAT_R16G16_UNORM	35	DXGI_FORMAT_R16G16_UNORM
DX_FORMAT_R16G16_UINT	36	DXGI_FORMAT_R16G16_UINT
DX_FORMAT_R16G16_SNORM	37	DXGI_FORMAT_R16G16_SNORM
DX_FORMAT_R16G16_SINT	38	DXGI_FORMAT_R16G16_SINT
DX_FORMAT_R32_TYPELESS	39	DXGI_FORMAT_R32_TYPELESS
DX_FORMAT_D32_FLOAT	40	DXGI_FORMAT_D32_FLOAT
DX_FORMAT_R32_FLOAT	41	DXGI_FORMAT_R32_FLOAT
DX_FORMAT_R32_UINT	42	DXGI_FORMAT_R32_UINT
DX_FORMAT_R32_SINT	43	DXGI_FORMAT_R32_SINT
DX_FORMAT_R24G8_TYPELESS	44	DXGI_FORMAT_R24G8_TYPELESS
DX_FORMAT_D24_UNORM_S8_UINT	45	DXGI_FORMAT_D24_UNORM_S8_UINT
DX_FORMAT_R24_UNORM_X8_TYPELESS	46	DXGI_FORMAT_R24_UNORM_X8_TYPELESS
DX_FORMAT_X24_TYPELESS_G8_UINT	47	DXGI_FORMAT_X24_TYPELESS_G8_UINT
DX_FORMAT_R8G8_TYPELESS	48	DXGI_FORMAT_R8G8_TYPELESS
DX_FORMAT_R8G8_UNORM	49	DXGI_FORMAT_R8G8_UNORM
DX_FORMAT_R8G8_UINT	50	DXGI_FORMAT_R8G8_UINT
DX_FORMAT_R8G8_SNORM	51	DXGI_FORMAT_R8G8_SNORM

© 2000-2024, MetaQuotes Ltd.

ID	Valeur	Correspondant au format DXGI_FORMAT
DX_FORMAT_R8G8_SINT	52	DXGI_FORMAT_R8G8_SINT
DX_FORMAT_R16_TYPELESS	53	DXGI_FORMAT_R16_TYPELESS
DX_FORMAT_R16_FLOAT	54	DXGI_FORMAT_R16_FLOAT
DX_FORMAT_D16_UNORM	55	DXGI_FORMAT_D16_UNORM
DX_FORMAT_R16_UNORM	56	DXGI_FORMAT_R16_UNORM
DX_FORMAT_R16_UINT	57	DXGI_FORMAT_R16_UINT
DX_FORMAT_R16_SNORM	58	DXGI_FORMAT_R16_SNORM
DX_FORMAT_R16_SINT	59	DXGI_FORMAT_R16_SINT
DX_FORMAT_R8_TYPELESS	60	DXGI_FORMAT_R8_TYPELESS
DX_FORMAT_R8_UNORM	61	DXGI_FORMAT_R8_UNORM
DX_FORMAT_R8_UINT	62	DXGI_FORMAT_R8_UINT
DX_FORMAT_R8_SNORM	63	DXGI_FORMAT_R8_SNORM
DX_FORMAT_R8_SINT	64	DXGI_FORMAT_R8_SINT
DX_FORMAT_A8_UNORM	65	DXGI_FORMAT_A8_UNORM
DX_FORMAT_R1_UNORM	66	DXGI_FORMAT_R1_UNORM
DX_FORMAT_R9G9B9E5_SHAREDEXP	67	DXGI_FORMAT_R9G9B9E5_SHAREDEXP
DX_FORMAT_R8G8_B8G8_UNORM	68	DXGI_FORMAT_R8G8_B8G8_UNORM
DX_FORMAT_G8R8_G8B8_UNORM	69	DXGI_FORMAT_G8R8_G8B8_UNORM
DX_FORMAT_BC1_TYPELESS	70	DXGI_FORMAT_BC1_TYPELESS
DX_FORMAT_BC1_UNORM	71	DXGI_FORMAT_BC1_UNORM
DX_FORMAT_BC1_UNORM_SRGB	72	DXGI_FORMAT_BC1_UNORM_SRGB
DX_FORMAT_BC2_TYPELESS	73	DXGI_FORMAT_BC2_TYPELESS
DX_FORMAT_BC2_UNORM	74	DXGI_FORMAT_BC2_UNORM
DX_FORMAT_BC2_UNORM_SRGB	75	DXGI_FORMAT_BC2_UNORM_SRGB
DX_FORMAT_BC3_TYPELESS	76	DXGI_FORMAT_BC3_TYPELESS
DX_FORMAT_BC3_UNORM	77	DXGI_FORMAT_BC3_UNORM
DX_FORMAT_BC3_UNORM_SRGB	78	DXGI_FORMAT_BC3_UNORM_SRGB
DX_FORMAT_BC4_TYPELESS	79	DXGI_FORMAT_BC4_TYPELESS
DX_FORMAT_BC4_UNORM	80	DXGI_FORMAT_BC4_UNORM
DX_FORMAT_BC4_SNORM	81	DXGI_FORMAT_BC4_SNORM

© 2000-2024, MetaQuotes Ltd.

ID	Valeur	Correspondant au format DXGI_FORMAT
DX_FORMAT_BC5_TYPELESS	82	DXGI_FORMAT_BC5_TYPELESS
DX_FORMAT_BC5_UNORM	83	DXGI_FORMAT_BC5_UNORM
DX_FORMAT_BC5_SNORM	84	DXGI_FORMAT_BC5_SNORM
DX_FORMAT_B5G6R5_UNORM	85	DXGI_FORMAT_B5G6R5_UNORM
DX_FORMAT_B5G5R5A1_UNORM	86	DXGI_FORMAT_B5G5R5A1_UNORM
DX_FORMAT_B8G8R8A8_UNORM	87	DXGI_FORMAT_B8G8R8A8_UNORM
DX_FORMAT_B8G8R8X8_UNORM	88	DXGI_FORMAT_B8G8R8X8_UNORM
DX_FORMAT_R10G10B10_XR_BIAS_A2_UNORM	89	DXGI_FORMAT_R10G10B10_XR_BIAS_A2_UNORM
DX_FORMAT_B8G8R8A8_TYPELESS	90	DXGI_FORMAT_B8G8R8A8_TYPELESS
DX_FORMAT_B8G8R8A8_UNORM_SRGB	91	DXGI_FORMAT_B8G8R8A8_UNORM_SRGB
DX_FORMAT_B8G8R8X8_TYPELESS	92	DXGI_FORMAT_B8G8R8X8_TYPELESS
DX_FORMAT_B8G8R8X8_UNORM_SRGB	93	DXGI_FORMAT_B8G8R8X8_UNORM_SRGB
DX_FORMAT_BC6H_TYPELESS	94	DXGI_FORMAT_BC6H_TYPELESS
DX_FORMAT_BC6H_UF16	95	DXGI_FORMAT_BC6H_UF16
DX_FORMAT_BC6H_SF16	96	DXGI_FORMAT_BC6H_SF16
DX_FORMAT_BC7_TYPELESS	97	DXGI_FORMAT_BC7_TYPELESS
DX_FORMAT_BC7_UNORM	98	DXGI_FORMAT_BC7_UNORM
DX_FORMAT_BC7_UNORM_SRGB	99	DXGI_FORMAT_BC7_UNORM_SRGB
DX_FORMAT_AYUV	100	DXGI_FORMAT_AYUV
DX_FORMAT_Y410	101	DXGI_FORMAT_Y410
DX_FORMAT_Y416	102	DXGI_FORMAT_Y416
DX_FORMAT_NV12	103	DXGI_FORMAT_NV12
DX_FORMAT_P010	104	DXGI_FORMAT_P010
DX_FORMAT_P016	105	DXGI_FORMAT_P016
DX_FORMAT_420_OPAQUE	106	DXGI_FORMAT_420_OPAQUE
DX_FORMAT_YUY2	107	DXGI_FORMAT_YUY2
DX_FORMAT_Y210	108	DXGI_FORMAT_Y210
DX_FORMAT_Y216	109	DXGI_FORMAT_Y216

© 2000-2024, MetaQuotes Ltd.

ID	Valeur	Correspondant au format DXGI_FORMAT
DX_FORMAT_NV11	110	DXGI_FORMAT_NV11
DX_FORMAT_AI44	111	DXGI_FORMAT_AI44
DX_FORMAT_IA44	112	DXGI_FORMAT_IA44
DX_FORMAT_P8	113	DXGI_FORMAT_P8
DX_FORMAT_A8P8	114	DXGI_FORMAT_A8P8
DX_FORMAT_B4G4R4A4_UNORM	115	DXGI_FORMAT_B4G4R4A4_UNORM
DX_FORMAT_P208	130	DXGI_FORMAT_P208
DX_FORMAT_V208	131	DXGI_FORMAT_V208
DX_FORMAT_V408	132	DXGI_FORMAT_V408
DX_FORMAT_FORCE_UINT	0xffffffff	DXGI_FORMAT_FORCE_UINT

© 2000-2024, MetaQuotes Ltd.

DXInputCreate

Crée les entrées d'un shader.

```
int  DXInputCreate(
   int   context,        // handle du contexte graphique
   uint  input_size      // taille des entrées en octets
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

input_size

[in] Taille de la structure des paramètres en octets.

Valeur de Retour

Le handle pour les entrées du shader ou INVALID_HANDLE en cas d'erreur. Pour obtenir le code d'erreur, utilisez la fonction GetLastError().

Un handle créé qui n'est plus utilisé doit être libéré explicitement avec la fonction DXRelease().

© 2000-2024, MetaQuotes Ltd.

DXInputSet

Définit les entrées d'un shader.

```
bool  DXInputSet(
   int          input,       // handle du contexte graphique
   const void&  data         // données à utiliser
   );
```

Paramètres

input

[in] Handle des entrées pour le shader, obtenu avec DXInputCreate().

data

[in] Données à utiliser pour les entrées du shader.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

DXShaderCreate

Crée un shader du type spécifié.

```
int  DXShaderCreate(
   int                  context,          // handle du contexte graphique
   ENUM_DX_SHADER_TYPE  shader_type,      // type du shader
   const string         source,           // code source du shader
   const string         entry_point,      // point d'entrée
   string&              compile_error     // message du compilateur en cas d'erreur
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

shader_type

[out] Une valeur de l'énumération ENUM_DX_SHADER_TYPE.

source

[in] Code source du shader en HLSL 5.

entry_point

[in] Point d'entrée - nom d'une fonction dans le code source.

compile_error

[in] Chaîne de caractères pour récupérer les erreurs de compilation.

Valeur de Retour

Handle du shader ou INVALID_HANDLE en cas d'erreur. Pour obtenir le code d'erreur, utilisez la fonction GetLastError().

Note

Un handle créé qui n'est plus utilisé doit être libéré explicitement avec la fonction DXRelease().

ENUM_DX_SHADER_TYPE

ID	Valeur	Description
DX_SHADER_VERTEX	0	Shader du sommet
DX_SHADER_GEOMETRY	1	Shader de la géométrie
DX_SHADER_PIXEL	2	Shader du pixel

© 2000-2024, MetaQuotes Ltd.

DXShaderSetLayout

Définit la mise en page du sommet pour le shader du sommet.

```
bool  DXShaderSetLayout(
   int                     shader,        // handle du shader
   const DXVertexLayout&   layout[]       //
   );
```

Paramètres

shader

[in] Handle du shader d'un sommet créé avec DXShaderCreate().

layout[]

[in] Tableau des descriptions des champs du sommet. La description est définie avec la structure DXVertexLayout :

```
struct DXVertexLayout
  {
   string         semantic_name;       // La sémantique HLSL associée avec cet élé
   uint           semantic_index;      // L'indice sémantique pour l'élément. Un i
   ENUM_DX_FORMAT format;              // le type de donnée de l'élément.
  };
```

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

La mise en page doit correspondre au type des sommets dans un buffer de sommet spécifié. Elle doit aussi correspondre au type d'entrée des sommets utilisés au point d'entrée dans le code du shader du sommet.

Le buffer de sommet d'un shader est défini avec DXBufferSet().

La structure DXVertexLayout est une version de la structure MSDN D3D11_INPUT_ELEMENT_DESC.

© 2000-2024, MetaQuotes Ltd.

DXShaderInputsSet

Définit les entrées d'un shader.

```
bool  DXShaderInputsSet(
   int         shader,       // handle du shader
   const int&  inputs[]      // tableau des handles d'entrée
   );
```

Paramètres

shader

[in] Handle du shader créé avec DXShaderCreate().

inputs[]

[in] Tableau de handles d'entrée créés avec DXInputCreate().

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

La taille des paramètres d'entrée doit être égale au nombre d'objets cbuffer déclarés dans le code du shader.

© 2000-2024, MetaQuotes Ltd.

DXShaderTexturesSet

Définit les textures du shader.

```
bool  DXShaderTexturesSet(
   int          shader,          // handle du shader
   const  int&  textures[]       // tableau de handles
   );
```

Paramètres

shader

[in] Handle du shader créé avec DXShaderCreate().

textures[]

[in] Tableau de handles de textures créés avec DXTextureCreate().

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

La taille du tableau de textures doit être égale au nombre d'objets Texture2D déclarés dans le code du shader.

© 2000-2024, MetaQuotes Ltd.

DXDraw

Effectue le rendering des sommets du buffer du sommet définit avec DXBufferSet().

```
bool  DXDraw(
   int   context,              // handle du contexte graphique
   uint  start=0,              // indice du premier sommet
   uint  count=WHOLE_ARRAY     // nombre de sommets
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

start=0

[in] Indice du premier sommet à dessiner.

count=WHOLE_ARRAY

[in] Nombre de sommets à dessiner.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

Les shaders doivent d'abord être définis en utilisant DXShaderSet() pour le rendu des sommets.

© 2000-2024, MetaQuotes Ltd.

DXDrawIndexed

Effectue le rendering des primitives graphiques décrites par le buffer d'index depuis DXBufferSet().

```
bool  DXDrawIndexed(
   int   context,               // handle du contexte graphique
   uint  start=0,               // indice de la première primitive
   uint  count=WHOLE_ARRAY      // nombre de primitives
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

start

[in] Indice de la première primitive pour le rendu.

count

[in] Nombre de primitives pour le rendu.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

Le type des primitives décrites par le buffer d'indices est défini avec DXPrimiveTopologySet().

Le buffer de sommets dans DXBufferSet() doit être défini au préalable pour effectuer le rendu des primitives.

Les shaders doivent également être définis avec DXShaderSet().

© 2000-2024, MetaQuotes Ltd.

DXPrimiveTopologySet

Définit le type des primitives du rendu en utilisant DXDrawIndexed().

```
bool  DXPrimiveTopologySet(
   int                          context,                // handle du contexte graphique
   ENUM_DX_PRIMITIVE_TOPOLOGY   primitive_topology      // type de primitive
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

primitive_topology

[in] Une valeur de l'énumération ENUM_DX_PRIMITIVE_TOPOLOGY.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

ENUM_DX_PRIMITIVE_TOPOLOGY

ID	Valeur	Correspondance dans D3D11_PRIMITIVE_TOPOLOGY
DX_PRIMITIVE_TOPOLOGY_POINTLIST	1	D3D11_PRIMITIVE_TOPOLOGY_POINTLIST
DX_PRIMITIVE_TOPOLOGY_LINELIST	2	D3D11_PRIMITIVE_TOPOLOGY_LINELIST
DX_PRIMITIVE_TOPOLOGY_LINESTRIP	3	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST	4	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP	5	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP
DX_PRIMITIVE_TOPOLOGY_LINELIST_ADJ	6	D3D11_PRIMITIVE_TOPOLOGY_LINELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ	7	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ	8	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ	9	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ

© 2000-2024, MetaQuotes Ltd.

DXBufferSet

Définit le buffer pour le rendu courant.

```
bool  DXBufferSet(
   int   context,              // handle du contexte graphique
   int   buffer,               // handle du sommet ou du buffer d'index
   uint  start=0,              // indice initial
   uint  count=WHOLE_ARRAY     // nombres d'éléments
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

buffer

[in] Handle du sommet ou du buffer d'index créé avec DXBufferCreate().

start

[in] Indice du premier élément du buffer. Les données sont utilisées par défaut depuis le début du buffer.

count

[in] Nombre de valeurs à utiliser. Par défaut, toutes les valeurs sont utilisées.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

La fonction DXBufferSet() doit être appelée pour définir le sommet et les buffers d'index pour le rendu en utilisant DXDraw().

© 2000-2024, MetaQuotes Ltd.

DXShaderSet

Définit un shader pour le rendu.

```
bool  DXShaderSet(
   int  context,       // handle du contexte graphique
   int  shader         // handle du shader
   );
```

Paramètres

context

[in] Handle du contexte graphique créé avec DXContextCreate().

shader

[in] Handle du shader créé avec DXShaderCreate().

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

Plusieurs types de shaders (sommet, géométrie et pixel) peuvent être utilisés simultanément pour le rendu.

© 2000-2024, MetaQuotes Ltd.

DXHandleType

Retourne le type du handle.

```
ENUM_DX_HANDLE_TYPE  DXHandleType(
   int  handle      // handle
   );
```

Paramètres

handle

[in] Handle.

Valeur de Retour

Une valeur de l'énumération ENUM_DX_HANDLE_TYPE

ENUM_DX_HANDLE_TYPE

ID	Valeur	Description
DX_HANDLE_INVALID	0	Handle invalide
DX_HANDLE_CONTEXT	1	Handle du contexte graphique
DX_HANDLE_SHADER	2	Handle du shader
DX_HANDLE_BUFFER	3	Handle du sommet ou du buffer d'indices
DX_HANDLE_INPUT	4	Handle pour les entrées du shader
DX_HANDLE_TEXTURE	5	Handle de la texture

© 2000-2024, MetaQuotes Ltd.

DXRelease

Libère un handle.

```
bool  DXRelease(
   int  handle      // handle
   );
```

Paramètres

context

[in] Handle libéré.

Valeur de Retour

Retourne true en cas d'exécution réussie avec succès, sinon false. Pour obtenir le code d'erreur, appelez la fonction GetLastError().

Note

Tous les handles créés et qui ne sont plus utilisés doivent être libérés explicitement avec la fonction DXRelease().

© 2000-2024, MetaQuotes Ltd.

Module MetaTrader pour l'intégration avec Python

MQL5 est conçu pour le développement d'applicatoins de trading à haute performance sur les marchés financiers et n'a pas d'équivalent parmi les autres langages spécialisés utilisés dans le trading algorithmique. La syntaxe et la vitesse des programmes MQL5 sont aussi proches que possible de ceux en C++. Il y a le support d'OpenCL et l'intégration avec MS Visual Studio. Des bibliothèques de statistiques, de logique floue et pour ALGLIB sont également disponibles. L'environnement de développement MetaEditor comporte le support natif des bibliothèques .NET avec l'import "intelligent" de fonctions, éliminant le besoin de développer des wrappers spéciaux. Les DLL C++ tierces peuvent également être utilisées. Les fichiers de code source C++ (CPP et H) peuvent être édités et compilés dans des DLL directement depuis l'éditeur. Microsoft Visual Studio installé sur le PC d'un utilisateur peut également être utilisé pour cela.

Python est un langage de programmation de haut niveau moderne pour développer des scripts et des applications. Il contient plusieurs bibliothèques pour le machine learning, l'automatisation des processus, ainsi que l'analyse et la visualisation des données.

Le package MetaTrader pour Python est conçu pour obtenir facilement et rapidement des données boursières via une communication interprocesseur directement à partir du terminal MetaTrader 5. Les données reçues de cette manière peuvent être utilisées pour des calculs statistiques et le machine learning.

Installation du package à partir de la ligne de commande :

```
pip install MetaTrader5
```

Mise à jour du package à partir de la ligne de commande :

```
pip install --upgrade MetaTrader5
```

Fonctions d'intégration de MetaTrader 5 et Python

Fonction	Action
initialize	Établit une connexion avec le terminal MetaTrader 5
login	Connexion à un compte de trading en utilisant les paramètres spécifiés
shutdown	Ferme la connexion précédemment établie avec le terminal MetaTrader 5
version	Retourne la version du terminal MetaTrader 5
last_error	Retourne les données de la dernière erreur
account_info	Retourne des informations sur le compte de trading actuel
terminal_Info	Retourne l'état et les paramètres du terminal MetaTrader 5 connecté
symbols_total	Retourne le nombre total d'instruments financiers dans le terminal MetaTrader 5
symbols_get	Retourne tous les instruments financiers du terminal MetaTrader 5
symbol_info	Retourne les données sur l'instrument financier spécifié

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
symbol_info_tick	Sélectionne un symbole dans le MarketWatch
symbol_select	Sélectionne un symbole dans la fenêtre MarketWatch ou supprime un symbole de la fenêtre
market_book_add	Abonne le terminal MetaTrader 5 aux événements de changements du Market Depth pour un symbole spécifié
market_book_get	Renvoie un tuple de BookInfo contenant des entrées du Market Depth pour le symbole spécifié
market_book_release	Annule l'abonnement du terminal MetaTrader 5 aux événements de changement de Market Depth pour un symbole spécifié
copy_rates_from	Retourne les barres du terminal MetaTrader 5 à partir de la date spécifiée
copy_rates_from_pos	Retourne les barres du terminal MetaTrader 5 à partir de l'index spécifié
copyrates_range	Retourne les barres du terminal MetaTrader 5 dans la plage de dates spécifiée
copy_ticks_from	Retourne les ticks du terminal MetaTrader 5 à partir de la date spécifiée
copy_ticks_range	Retourne les ticks du terminal MetaTrader 5 pour la plage de dates spécifiée
orders_total	Retourne le nombre d'ordre actifs.
orders_get	Retourne les ordres actifs avec la possibilité de filtrer par symbole ou ticket
order_calc_margin	Retourne la marge dans la devise du compte pour effectuer une opération de trading spécifiée
order_calc_profit	Retourne le profit dans la devise du compte pour une opération de trading spécifiée
order_check	Vérifie la suffisance des fonds pour effectuer une opération de trading désirée
order_send	Envoie une requête pour effectuer une opération de trading.
positions_total	Retourne le nombre de positions ouvertes
positions_get	Retourne les positions ouvertes avec la possibilité de filtrer par symbole ou ticket
history_orders_total	Retourne le nombre d'ordres dans l'historique de trading dans l'intervalle spécifié
history_orders_get	Retourne les ordres de l'historique de trading avec la possibilité de filtrer par ticket ou par position

© 2000-2024, MetaQuotes Ltd.

Fonction	Action
history_deals_total	Retourne le nombre de transactions dans l'historique de trading dans l'intervalle spécifié
history_deals_get	Retourne les transactions de l'historique de trading avec la possibilité de filtrer par ticket ou par position

Exemple de connexion de Python à MetaTrader 5

1. Téléchargez la dernière version de Python 3.8 depuis https://www.python.org/downloads/windows
2. Lors de l'installation de Python, cochez "Ajouter Python 3.8 au PATH%" pour pouvoir exécuter des scripts Python à partir de la ligne de commande.
3. Installez le module MetaTrader 5 à partir de la ligne de commande

```
pip install MetaTrader5
```

4. Ajoutez les packages matplotlib et pandas

```
pip install matplotlib
pip install pandas
```

5. Lancez le script de test

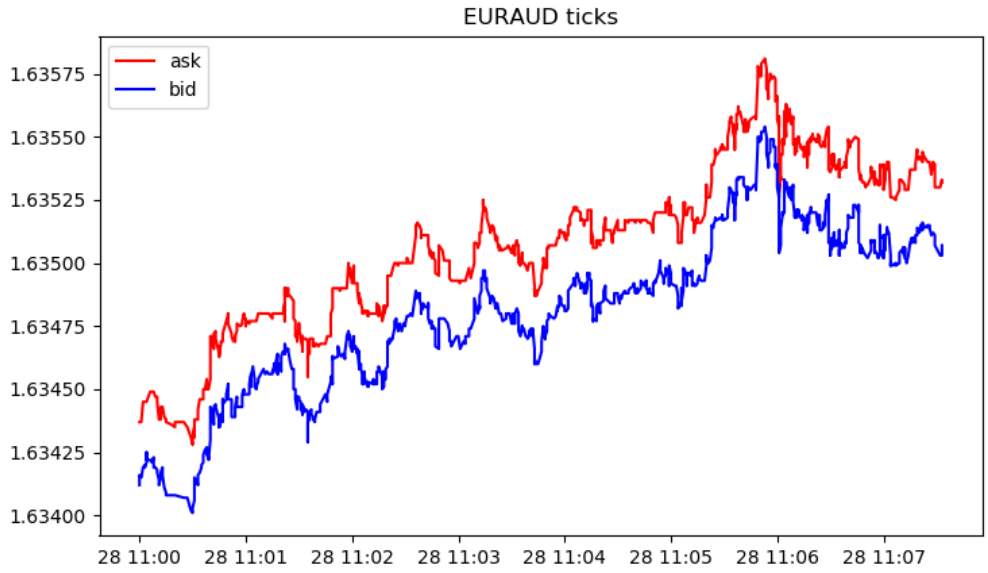
```
from datetime import datetime
import matplotlib.pyplot as plt
import pandas as pd
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import MetaTrader5 as mt5

# connexion à MetaTrader 5
if not mt5.initialize():
    print("initialize() a échoué")
    mt5.shutdown()

# demande l'état et les paramètres de connexion
print(mt5.terminal_info())
# récupère les données sur la version MetaTrader 5
print(mt5.version())

# demande 1000 ticks à EURAUD
euraud_ticks = mt5.copy_ticks_from("EURAUD", datetime(2020,1,28,13), 1000, mt5.COPY_TI
# demande les ticks de AUDUSD du 2019.04.01 13:00 au 2019.04.02 13:00
audusd_ticks = mt5.copy_ticks_range("AUDUSD", datetime(2020,1,27,13), datetime(2020,1,

# récupère les barres de différents symboles de différentes manières
eurusd_rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_M1, datetime(2020,1,28,13),
eurgbp_rates = mt5.copy_rates_from_pos("EURGBP", mt5.TIMEFRAME_M1, 0, 1000)
eurcad_rates = mt5.copy_rates_range("EURCAD", mt5.TIMEFRAME_M1, datetime(2020,1,27,13)

# ferme la connexion à MetaTrader 5
```

© 2000-2024, MetaQuotes Ltd.

```
mt5.shutdown()

#DATA
print('euraud_ticks(', len(euraud_ticks), ')')
for val in euraud_ticks[:10]: print(val)

print('audusd_ticks(', len(audusd_ticks), ')')
for val in audusd_ticks[:10]: print(val)

print('eurusd_rates(', len(eurusd_rates), ')')
for val in eurusd_rates[:10]: print(val)

print('eurgbp_rates(', len(eurgbp_rates), ')')
for val in eurgbp_rates[:10]: print(val)

print('eurcad_rates(', len(eurcad_rates), ')')
for val in eurcad_rates[:10]: print(val)

#DESSIN
# crée le DataFrame à partir des données obtenues
ticks_frame = pd.DataFrame(euraud_ticks)
# convertit le temps en secondes au format datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')
# affiche les ticks sur le graphique
plt.plot(ticks_frame['time'], ticks_frame['ask'], 'r-', label='ask')
plt.plot(ticks_frame['time'], ticks_frame['bid'], 'b-', label='bid')

# affiche les légendes
plt.legend(loc='upper left')

# ajoute l'en-tête
plt.title('EURAUD ticks')

# affiche le graphique
plt.show()
```

© 2000-2024, MetaQuotes Ltd.

6. Récupère les données et le graphique




```
[2, 'MetaQuotes-Demo', '16167573']
[500, 2325, '19 Feb 2020']

euraud_ticks( 1000 )
(1580209200, 1.63412, 1.63437, 0., 0, 1580209200067, 130, 0.)
(1580209200, 1.63416, 1.63437, 0., 0, 1580209200785, 130, 0.)
(1580209201, 1.63415, 1.63437, 0., 0, 1580209201980, 130, 0.)
(1580209202, 1.63419, 1.63445, 0., 0, 1580209202192, 134, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203004, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203487, 130, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203694, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203990, 130, 0.)
(1580209204, 1.63421, 1.63445, 0., 0, 1580209204194, 130, 0.)
(1580209204, 1.63425, 1.63445, 0., 0, 1580209204392, 130, 0.)
audusd_ticks( 40449 )
(1580122800, 0.67858, 0.67868, 0., 0, 1580122800244, 130, 0.)
(1580122800, 0.67858, 0.67867, 0., 0, 1580122800429, 4, 0.)
(1580122800, 0.67858, 0.67865, 0., 0, 1580122800817, 4, 0.)
(1580122801, 0.67858, 0.67866, 0., 0, 1580122801618, 4, 0.)
(1580122802, 0.67858, 0.67865, 0., 0, 1580122802928, 4, 0.)
(1580122809, 0.67855, 0.67865, 0., 0, 1580122809526, 130, 0.)
(1580122809, 0.67855, 0.67864, 0., 0, 1580122809699, 4, 0.)
(1580122813, 0.67855, 0.67863, 0., 0, 1580122813576, 4, 0.)
(1580122815, 0.67856, 0.67863, 0., 0, 1580122815190, 130, 0.)
(1580122815, 0.67855, 0.67863, 0., 0, 1580122815479, 130, 0.)
eurusd_rates( 1000 )
(1580149260, 1.10132, 1.10151, 1.10131, 1.10149, 44, 1, 0)
(1580149320, 1.10149, 1.10161, 1.10143, 1.10154, 42, 1, 0)
(1580149380, 1.10154, 1.10176, 1.10154, 1.10174, 40, 2, 0)
(1580149440, 1.10174, 1.10189, 1.10168, 1.10187, 47, 1, 0)
```

© 2000-2024, MetaQuotes Ltd.

```
(1580149500, 1.10185, 1.10191, 1.1018, 1.10182, 53, 1, 0)
(1580149560, 1.10182, 1.10184, 1.10176, 1.10183, 25, 3, 0)
(1580149620, 1.10183, 1.10187, 1.10177, 1.10187, 49, 2, 0)
(1580149680, 1.10187, 1.1019, 1.1018, 1.10187, 53, 1, 0)
(1580149740, 1.10187, 1.10202, 1.10187, 1.10198, 28, 2, 0)
(1580149800, 1.10198, 1.10198, 1.10183, 1.10188, 39, 2, 0)
eurgbp_rates( 1000 )
(1582236360, 0.83767, 0.83767, 0.83764, 0.83765, 23, 9, 0)
(1582236420, 0.83765, 0.83765, 0.83764, 0.83765, 15, 8, 0)
(1582236480, 0.83765, 0.83766, 0.83762, 0.83765, 19, 7, 0)
(1582236540, 0.83765, 0.83768, 0.83758, 0.83763, 39, 6, 0)
(1582236600, 0.83763, 0.83768, 0.83763, 0.83767, 21, 6, 0)
(1582236660, 0.83767, 0.83775, 0.83765, 0.83769, 63, 5, 0)
(1582236720, 0.83769, 0.8377, 0.83758, 0.83764, 40, 7, 0)
(1582236780, 0.83766, 0.83769, 0.8376, 0.83766, 37, 6, 0)
(1582236840, 0.83766, 0.83772, 0.83763, 0.83772, 22, 6, 0)
(1582236900, 0.83772, 0.83773, 0.83768, 0.8377, 36, 5, 0)
eurcad_rates( 1441 )
(1580122800, 1.45321, 1.45329, 1.4526, 1.4528, 146, 15, 0)
(1580122860, 1.4528, 1.45315, 1.45274, 1.45301, 93, 15, 0)
(1580122920, 1.453, 1.45304, 1.45264, 1.45264, 82, 15, 0)
(1580122980, 1.45263, 1.45279, 1.45231, 1.45277, 109, 15, 0)
(1580123040, 1.45275, 1.4528, 1.45259, 1.45271, 53, 14, 0)
(1580123100, 1.45273, 1.45285, 1.45269, 1.4528, 62, 16, 0)
(1580123160, 1.4528, 1.45284, 1.45267, 1.45282, 64, 14, 0)
(1580123220, 1.45282, 1.45299, 1.45261, 1.45272, 48, 14, 0)
(1580123280, 1.45272, 1.45275, 1.45255, 1.45275, 74, 14, 0)
(1580123340, 1.45275, 1.4528, 1.4526, 1.4528, 94, 13, 0)
```

© 2000-2024, MetaQuotes Ltd.

initialize

Établit une connexion avec le terminal MetaTrader 5. Il existe trois options d'appel.

Appel sans paramètres. Le terminal de connexion est trouvé automatiquement.

```
initialize()
```

Appel spécifiant le chemin d'accès au terminal MetaTrader 5 auquel nous voulons nous connecter.

```
initialize(
   path                      // chemin vers le fichier EXE du terminal MetaTrader 5
   )
```

Appel en spécifiant le chemin et les paramètres du compte de trading.

```
initialize(
   path,                     // chemin vers le fichier EXE du terminal MetaTrader 5
   login=LOGIN,              // numéro du compte
   password="PASSWORD",      // mot de passe
   server="SERVER",          // nom du serveur tel qu'il est spécifié dans le terminal
   timeout=TIMEOUT,          // timeout
   portable=False            // mode portable
   )
```

Paramètres

path

[in] Chemin d'accès au fichier metatrader.exe ou metatrader64.exe. Paramètre sans nom facultatif. Il est indiqué en premier sans nom de paramètre. Si le chemin n'est pas spécifié, le module tente de trouver le fichier exécutable par lui-même.

login=LOGIN

[in] Numéro du compte de trading. Paramètre nommé facultatif. S'il n'est pas spécifié, le dernier compte de trading est utilisé.

password="PASSWORD"

[in] Mot de passe du compte de trading. Paramètre nommé facultatif. Si le mot de passe n'est pas défini, le mot de passe d'un compte de trading spécifié enregistré dans la base de données du terminal est appliqué automatiquement.

server="SERVER"

[in] Nom du serveur de trading. Paramètre nommé facultatif. Si le serveur n'est pas défini, le serveur d'un compte de trading spécifié enregistré dans la base de données du terminal est appliqué automatiquement.

timeout=TIMEOUT

[in] Délai d'expiration de la connexion en millisecondes. Paramètre nommé facultatif. S'il n'est pas spécifié, la valeur de 60 000 (60 secondes) est appliquée.

portable=False

[in] Flag du lancement du terminal en mode portable. Paramètre nommé facultatif. S'il n'est pas spécifié, la valeur False est utilisée.

© 2000-2024, MetaQuotes Ltd.

Valeur de Retour

Renvoie True en cas de connexion réussie au terminal MetaTrader 5, sinon - False.

Note

Si nécessaire, le terminal MetaTrader 5 est lancé pour établir la connexion lors de l'exécution de l'appel initialize().

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion MetaTrader 5 à un compte de trading spécifié
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# affiche les données sur l'état de la connexion, le nom du serveur et le compte de tr
print(mt5.terminal_info())
# affiche les données sur la version de MetaTrader 5
print(mt5.version())

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

Voir aussi

shutdown, terminal_info, version

© 2000-2024, MetaQuotes Ltd.

login

Connexion à un compte de trading en utilisant les paramètres spécifiés.

```
login(
   login,                    // numéro du compte
   password="PASSWORD",      // mot de passe
   server="SERVER",          // nom du serveur tel que spécifié dans le terminal
   timeout=TIMEOUT           // timeout
   )
```

Paramètres

login

[in] Numéro du compte de trading. Paramètre non nommé requis.

password

[in] Mot de passe du compte de trading. Paramètre nommé facultatif. Si le mot de passe n'est pas défini, le mot de passe enregistré dans la base de données du terminal est appliqué automatiquement.

server

[in] Nom du serveur de trading. Paramètre nommé facultatif. Si aucun serveur n'est défini, le dernier serveur utilisé est appliqué automatiquement.

timeout=TIMEOUT

[in] Délai d'expiration de la connexion en millisecondes. Paramètre nommé facultatif. S'il n'est pas spécifié, la valeur de 60 000 (60 secondes) est appliquée. Si la connexion n'est pas établie dans le délai spécifié, l'appel est interrompu de force et l'exception est générée.

Valeur de Retour

True en cas de connexion réussie au compte de trading, sinon - False.

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# affiche les données sur la version de MetaTrader 5
print(mt5.version())
# se connecte au compte de trading sans spécifier de mot de passe et de serveur
account=17221085
authorized=mt5.login(account)  # le mot de passe de la base de données du terminal est
if authorized:
    print("connecté au compte #{}".format(account))
```

© 2000-2024, MetaQuotes Ltd.

```
else:
    print("Echec de la connexion au compte #{}, code d'erreur : {}".format(account, mt

# se connecte maintenant à un autre compte de trading en spécifiant le mot de passe
account=25115284
authorized=mt5.login(account, password="gqrtz0lbdm")
if authorized:
    # affiche les données brutes du compte de trading
    print(mt5.account_info())
    # affiche les données du compte de trading sous forme de liste
    print("Affiche account_info()._asdict():")
    account_info_dict = mt5.account_info()._asdict()
    for prop in account_info_dict:
        print("  {}={}".format(prop, account_info_dict[prop]))
else:
    print("Echec de la connexion au compte #{}, code d'erreur : {}".format(account, mt

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

```
Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29
[500, 2367, '23 Mar 2020']

connecté au compte #17221085

connecté au compte # 25115284
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
propriétés du compte :
   login=25115284
   trade_mode=0
   leverage=100
   limit_orders=200
   margin_so_mode=0
   trade_allowed=True
   trade_expert=True
   margin_mode=2
   currency_digits=2
   fifo_close=False
   balance=99588.33
   credit=0.0
   profit=-45.23
   equity=99543.1
   margin=54.37
   margin_free=99488.73
   margin_level=183084.6054809638
   margin_so_call=50.0
```

© 2000-2024, MetaQuotes Ltd.

```
    margin_so_so=30.0
    margin_initial=0.0
    margin_maintenance=0.0
    assets=0.0
    liabilities=0.0
    commission_blocked=0.0
    name=James Smith
    server=MetaQuotes-Demo
    currency=USD
    company=MetaQuotes Software Corp.
```

Voir aussi

initialize, shutdown

© 2000-2024, MetaQuotes Ltd.

shutdown

Ferme la connexion précédemment établie avec le terminal MetaTrader 5.

```
shutdown()
```

Valeur de Retour

None.

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() a échoué")
    quit()

# affiche les données sur l'état de la connexion, le nom du serveur et le compte de tr
print(mt5.terminal_info())
# affiche les données sur la version de MetaTrader 5
print(mt5.version())

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

Voir aussi

initialize, login_py, terminal_info, version

© 2000-2024, MetaQuotes Ltd.

version

Retourne la version du terminal MetaTrader 5.

```
version()
```

Valeur de Retour

Retourne la version du terminal MetaTrader 5, la date de construction et de sortie. Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

La fonction version() renvoie la version du terminal, la date de génération et de publication sous la forme d'un tuple de trois valeurs :

Type	Description	Exemple de valeur
integer	Version du terminal MetaTrader 5	500
integer	Build	2007
string	Date de sortie du build	'25 Feb 2019'

Exemple :

```
import MetaTrader5 as mt5
import pandas as pd
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# affiche les données sur la version de MetaTrader 5
print(mt5.version())

# affiche les données brutes sur l'état de la connexion, le nom du serveur et le compt
print(mt5.terminal_info())
print()

# obtient les propriétés sous la forme d'un dictionnaire
terminal_info_dict=mt5.terminal_info()._asdict()
# convertit le dictionnaire en DataFrame et l'imprime
df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
print("terminal_info() sous la forme de dataframe :")
print(df[:-1])
```

© 2000-2024, MetaQuotes Ltd.

```
# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()

Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29
[500, 2367, '23 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True, dlls_a

terminal_info() as dataframe:
                propriété                       valeur
0       community_account                         True
1    community_connection                         True
2               connected                         True
3            dlls_allowed                        False
4           trade_allowed                        False
5       tradeapi_disabled                        False
6           email_enabled                        False
7             ftp_enabled                        False
8   notifications_enabled                        False
9                    mqid                        False
10                  build                         2367
11                maxbars                         5000
12               codepage                         1251
13              ping_last                        77881
14      community_balance                      707.107
15         retransmission                            0
16                company     MetaQuotes Software Corp.
17                   name                  MetaTrader 5
18               language                      Russian
19                   path  E:\ProgramFiles\MetaTrader 5
20              data_path  E:\ProgramFiles\MetaTrader 5
```

Voir aussi

initialize, shutdown, terminal_info

© 2000-2024, MetaQuotes Ltd.

last_error

Renvoie des données sur la dernière erreur.

```
last_error()
```

Valeur de Retour

Renvoie le dernier code d'erreur et la description sous forme de tuple.

Note

last_error() permet d'obtenir un code d'erreur en cas d'échec de l'exécution d'une fonction d'une bibliothèque MetaTrader 5. Il est similaire à GetLastError(). Cependant, il applique ses propres codes d'erreur. Valeurs possibles :

Constante	Valeur	Description
RES_S_OK	1	succès générique
RES_E_FAIL	-1	échec générique
RES_E_INVALID_PARAMS	-2	arguments/aramètres invalides
RES_E_NO_MEMORY	-3	aucune condition de mémoire
RES_E_NOT_FOUND	-4	pas d'historique
RES_E_INVALID_VERSION	-5	version invalide
RES_E_AUTH_FAILED	-6	l'autorisation à échouée
RES_E_UNSUPPORTED	-7	méthode non supportée
RES_E_AUTO_TRADING_DISABLED	-8	trading automatique désactivé
RES_E_INTERNAL_FAIL	-10000	erreur générale IPC interne
RES_E_INTERNAL_FAIL_SEND	-10001	échec de l'envoi IPC interne
RES_E_INTERNAL_FAIL_RECEIVE	-10002	échec de réception IPC interne
RES_E_INTERNAL_FAIL_INIT	-10003	échec d'initialisation IPC interne
RES_E_INTERNAL_FAIL_CONNECT	-10003	IPC interne sans ipc
RES_E_INTERNAL_FAIL_TIMEOUT	-10005	délai d'expiration interne

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
```

© 2000-2024, MetaQuotes Ltd.

```
    quit()

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

Voir aussi

version, GetLastError

© 2000-2024, MetaQuotes Ltd.

account_info

Retourne des informations sur le compte de trading actuel.

```
account_info()
```

Valeur de Retour

Renvoie les informations sous la forme d'une structure de tuple nommée (namedtuple). Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

La fonction renvoie toutes les données qui peuvent être obtenues en utilisant AccountInfoInteger, AccountInfoDouble et AccountInfoString en un seul appel.

Exemple :

```
import MetaTrader5 as mt5
import pandas as pd
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# connexion au compte de trading en spécifiant un mot de passe et un serveur
authorized=mt5.login(25115284, password="gqz03431bdm")
if authorized:
    account_info=mt5.account_info()
    if account_info!=None:
        # affiche les données brutes du compte de trading
        print(account_info)
        # affiche les données du compte de trading sous la forme d'un dictionnaire
        print("Affiche account_info()._asdict():")
        account_info_dict = mt5.account_info()._asdict()
        for prop in account_info_dict:
            print("  {}={}".format(prop, account_info_dict[prop]))
        print()

       # convertit le dictionnaire en DataFrame et l'imprime
        df=pd.DataFrame(list(account_info_dict.items()),columns=['property','value'])
        print("account_info() en tableau :")
        print(df)
else:
    print("échec de la connexion au compte de trading 25115284 avec le mot de passe=gc

# ferme la connexion au terminal MetaTrader 5
```

© 2000-2024, MetaQuotes Ltd.

```
mt5.shutdown()


Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
Affiche account_info()._asdict():
  login=25115284
  trade_mode=0
  leverage=100
  limit_orders=200
  margin_so_mode=0
  trade_allowed=True
  trade_expert=True
  margin_mode=2
  currency_digits=2
  fifo_close=False
  balance=99511.4
  credit=0.0
  profit=41.82
  equity=99553.22
  margin=98.18
  margin_free=99455.04
  margin_level=101398.67590140559
  margin_so_call=50.0
  margin_so_so=30.0
  margin_initial=0.0
  margin_maintenance=0.0
  assets=0.0
  liabilities=0.0
  commission_blocked=0.0
  server=MetaQuotes-Demo
  currency=USD
  company=MetaQuotes Software Corp.

account_info() as dataframe
              property                     value
0                login                  25115284
1           trade_mode                         0
2             leverage                       100
3         limit_orders                       200
4       margin_so_mode                         0
5        trade_allowed                      True
6         trade_expert                      True
7          margin_mode                         2
8      currency_digits                         2
9           fifo_close                     False
10             balance                   99588.3
```

```
11              credit                          0
12              profit                     -45.13
13              equity                    99543.2
14              margin                      54.37
15         margin_free                    99488.8
16        margin_level                     183085
17      margin_so_call                         50
18        margin_so_so                         30
19      margin_initial                          0
20  margin_maintenance                          0
21              assets                          0
22         liabilities                          0
23  commission_blocked                          0
24                name                James Smith
25              server            MetaQuotes-Demo
26            currency                        USD
27             company  MetaQuotes Software Corp.
```

Voir aussi

initialize, shutdown, login

© 2000-2024, MetaQuotes Ltd.

terminal_info

Retourne l'état et les paramètres du terminal client MetaTrader 5 connecté.

```
terminal_info()
```

Valeur de Retour

Renvoie les informations sous la forme d'une structure de tuple nommée (namedtuple). Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

La fonction renvoie toutes les données qui peuvent être obtenues en utilisant TerminalInfoInteger, TerminalInfoDouble et _TerminalInfoDouble en un seul appel.

Exemple :

```
import MetaTrader5 as mt5
import pandas as pd
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# affiche les données sur la version de MetaTrader 5
print(mt5.version())
# affiche des informations sur les paramètres et l'état du terminal
terminal_info=mt5.terminal_info()
if terminal_info!=None:
    # affiche les données brutes du terminal
    print(terminal_info)
 # affiche les données sous forme de liste
    print("Affiche terminal_info()._asdict():")
    terminal_info_dict = mt5.terminal_info()._asdict()
    for prop in terminal_info_dict:
        print("  {}={}".format(prop, terminal_info_dict[prop]))
    print()
   # convertit le dictionnaire en DataFrame et l'imprime
    df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
    print("terminal_info() sous la forme de dataframe :")
    print(df)

# ferme la connexion au terminal MetaTrader 5
```

© 2000-2024, MetaQuotes Ltd.

```
mt5.shutdown()


Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29
[500, 2366, '20 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True,....
Affiche terminal_info()._asdict():
  community_account=True
  community_connection=True
  connected=True
  dlls_allowed=False
  trade_allowed=False
  tradeapi_disabled=False
  email_enabled=False
  ftp_enabled=False
  notifications_enabled=False
  mqid=False
  build=2366
  maxbars=5000
  codepage=1251
  ping_last=77850
  community_balance=707.10668201585
  retransmission=0.0
  company=MetaQuotes Software Corp.
  name=MetaTrader 5
  language=Russian
  path=E:\ProgramFiles\MetaTrader 5
  data_path=E:\ProgramFiles\MetaTrader 5
  commondata_path=C:\Users\Rosh\AppData\Roaming\MetaQuotes\Terminal\Common

terminal_info() as dataframe:
                 property                      value
0       community_account                       True
1    community_connection                       True
2               connected                       True
3            dlls_allowed                      False
4           trade_allowed                      False
5       tradeapi_disabled                      False
6           email_enabled                      False
7             ftp_enabled                      False
8   notifications_enabled                      False
9                    mqid                      False
10                  build                       2367
11                maxbars                       5000
12               codepage                       1251
13              ping_last                      80953
14      community_balance                    707.107
```

```
15         retransmission                   0.063593
16                company  MetaQuotes Software Corp.
17                   name                MetaTrader 5
18               language                     Russian
```

Voir aussi

initialize, shutdown, version

© 2000-2024, MetaQuotes Ltd.

symbols_total

Retourne le nombre total d'instruments financiers dans le terminal MetaTrader 5.

```
symbols_total()
```

Valeur de Retour

Valeur entière.

Note

La fonction est similaire à SymbolsTotal(). Cependant, il renvoie le nombre de tous les symboles, y compris les symboles personnalisés et ceux désactivés dans le MarketWatch.

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# récupère le nombre d'instruments financiers
symbols=mt5.symbols_total()
if symbols>0:
    print("Total de symboles =",symbols)
else:
    print("symboles non trouvés")

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

Voir aussi

symbols_get, symbol_select, symbol_info

© 2000-2024, MetaQuotes Ltd.

symbols_get

Retourne tous les instruments financiers du terminal MetaTrader 5.

```
symbols_get(
   group="GROUP"      // filtre de sélection du symbole
)
```

group="GROUP"

[in] Le filtre pour organiser un groupe de symboles nécessaires. Paramètre facultatif. Si le groupe est spécifié, la fonction renvoie uniquement les symboles répondant à des critères spécifiés.

Valeur de Retour

Renvoie les symboles sous la forme d'un tuple. Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

Le paramètre *group* permet de trier les symboles par nom. '*' peut être utilisé au début et à la fin d'une chaîne.

Le paramètre *group* peut être utilisé en tant que paramètre nommé ou non. Les deux options fonctionnent de la même manière. L'option nommée (group="GROUP") facilite la lecture du code.

Le paramètre *group* peut contenir plusieurs conditions séparées par des virgules. Une condition peut être définie comme un masque à l'aide de '*'. Le symbole de négation logique '!' peut être utilisé pour une exclusion. Toutes les conditions sont appliquées séquentiellement, ce qui signifie que les conditions d'inclusion à un groupe doivent être spécifiées en premier, suivies d'une condition d'exclusion. Par exemple, *group*= "*, !EUR" signifie que tous les symboles doivent être sélectionnés en premier et que ceux contenant "EUR" dans leur nom doivent être exclus par la suite.

Contrairement à symbol_info(), la fonction symbols_get() renvoie des données sur tous les symboles demandés dans un seul appel.

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# récupère tous les symboles
symbols=mt5.symbols_get()
print('Symboles : ', len(symbols))
count=0
# affiche les 5 premiers
for s in symbols:
```

© 2000-2024, MetaQuotes Ltd.

```
    count+=1
    print("{}. {}".format(count,s.name))
    if count==5: break
print()

# obtient les symboles contenant RU dans leurs noms
ru_symbols=mt5.symbols_get("*RU*")
print('len(*RU*): ', len(ru_symbols))
for s in ru_symbols:
    print(s.name)
print()

# obtient les symboles dont les noms ne contiennent pas USD, EUR, JPY et GBP
group_symbols=mt5.symbols_get(group="*,!*USD*,!*EUR*,!*JPY*,!*GBP*")
print('len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):', len(group_symbols))
for s in group_symbols:
    print(s.name,":",s)

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()

Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29
Symboles :  84
1. EURUSD
2. GBPUSD
3. USDCHF
4. USDJPY
5. USDCNH

len(*RU*):  8
EURUSD
USDRUB
USDRUR
EURRUR
EURRUB
FORTS.RUB.M5
EURUSD_T20
EURUSD4

len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):  13
AUDCAD : SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_dea
AUDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_
AUDNZD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_
CADCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_
NZDCAD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_
NZDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_
NZDSGD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_
```

© 2000-2024, MetaQuotes Ltd.

```
CADMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
CHFMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
FORTS.RTS.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
FORTS.RUB.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
FOREX.CHF.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
```

Voir aussi

symbols_total, symbol_select, symbol_info

© 2000-2024, MetaQuotes Ltd.

symbol_info

Retourne les données sur l'instrument financier spécifié.

```
symbol_info(
   symbol       // nom de l'instrument financier
)
```

symbol

[in] Nom de l'instrument financier. Paramètre non nommé requis.

Valeur de Retour

Renvoie les informations sous la forme d'une structure de tuple nommée (namedtuple). Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

La fonction renvoie toutes les données qui peuvent être obtenues en utilisant SymbolInfoInteger, SymbolInfoDouble et SymbolInfoString en un seul appel.

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# tentative d'activation de l'affichage du symbole EURJPY dans le MarketWatch
selected=mt5.symbol_select("EURJPY",True)
if not selected:
    print("Echec de la sélection de EURJPY")
    mt5.shutdown()
    quit()

# afficher les propriétés du symbole EURJPY
symbol_info=mt5.symbol_info("EURJPY")
if symbol_info!=None:
 # affiche les données brutes du terminal
    print(symbol_info)
    print("EURJPY: spread =",symbol_info.spread,"  digits =",symbol_info.digits)
 # affiche les propriétés des symboles sous forme de liste
    print("Affiche symbol_info(\"EURJPY\")._asdict():")
    symbol_info_dict = mt5.symbol_info("EURJPY")._asdict()
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
```

© 2000-2024, MetaQuotes Ltd.

```
# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()


Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0, ses
EURJPY: spread = 17   digits = 3
Affiche symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585069682
  digits=3
  spread=17
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
  option_mode=0
  option_right=0
  bid=120.024
  bidhigh=120.506
  bidlow=118.798
  ask=120.041
  askhigh=120.526
  asklow=118.828
  last=0.0
```

© 2000-2024, MetaQuotes Ltd.

```
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=0.001
trade_tick_value=0.8977708350166538
trade_tick_value_profit=0.8977708350166538
trade_tick_value_loss=0.8978272580355541
trade_tick_size=0.001
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-0.2
swap_short=-1.2
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
price_greeks_gamma=0.0
price_greeks_vega=0.0
price_greeks_rho=0.0
price_greeks_omega=0.0
price_sensitivity=0.0
basis=
category=
currency_base=EUR
currency_profit=JPY
```

© 2000-2024, MetaQuotes Ltd.

```
currency_margin=EUR
bank=
description=Euro vs Japanese Yen
exchange=
formula=
isin=
name=EURJPY
page=http://www.google.com/finance?q=EURJPY
path=Forex\EURJPY
```

Voir aussi

account_info, terminal_info

© 2000-2024, MetaQuotes Ltd.

symbol_info_tick

Retourne le dernier tick de l'instrument financier spécifié.

```
symbol_info_tick(
   symbol      // nom de l'instrument financier
)
```

symbol

[in] Nom de l'instrument financier. Paramètre non nommé requis.

Valeur de Retour

Retourne les informations sous la forme d'un tuple. Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

La fonction est similaire à SymbolInfoTick.

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# tentative d'activation de l'affichage du GBPUSD dans le MarketWatch
selected=mt5.symbol_select("GBPUSD",True)
if not selected:
    print("Echec de la sélection de GBPUSD")
    mt5.shutdown()
    quit()

# affiche le dernider tick GBPUSD
lasttick=mt5.symbol_info_tick("GBPUSD")
print(lasttick)
# affiche les valeurs des champs des ticks sous forme de liste
print("Affiche symbol_info_tick(\"GBPUSD\")._asdict():")
symbol_info_tick_dict = mt5.symbol_info_tick("GBPUSD")._asdict()
for prop in symbol_info_tick_dict:
    print("  {}={}".format(prop, symbol_info_tick_dict[prop]))

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

© 2000-2024, MetaQuotes Ltd.

```
Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29
Tick(time=1585070338, bid=1.17264, ask=1.17279, last=0.0, volume=0, time_msc=158507033
Affiche symbol_info_tick._asdict():
  time=1585070338
  bid=1.17264
  ask=1.17279
  last=0.0
  volume=0
  time_msc=1585070338728
  flags=2
  volume_real=0.0
```

Voir aussi

symbol_info, symbol_info

© 2000-2024, MetaQuotes Ltd.

symbol_select

Sélectionne un symbole dans la fenêtre MarketWatch ou supprime un symbole de la fenêtre.

```
symbol_select(
   symbol,       // nom de l'instrument financier
   enable=None   // active ou désactive
)
```

symbol

[in] Nom de l'instrument financier. Paramètre non nommé requis.

enable

[in] Switch. Paramètre sans nom facultatif. Si 'false', le symbole doit être supprimé de la fenêtre MarketWatch. Sinon, il doit être sélectionné dans la fenêtre MarketWatch. Un symbole ne peut pas être supprimé si des graphiques ouverts avec ce symbole sont actuellement présents ou si des positions y sont ouvertes.

Valeur de Retour

True en cas de succès, sinon - False.

Note

La fonction est similaire à SymbolSelect.

Exemple :

```
import MetaTrader5 as mt5
import pandas as pd
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)
print()
# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# tentative d'activation de l'affichage de l'EURCAD dans MarketWatch
selected=mt5.symbol_select("EURCAD",True)
if not selected:
 print("Impossible de sélectionner EURCAD, code d'erreur=",mt5.last_error())
else:
    symbol_info=mt5.symbol_info("EURCAD")
    print(symbol_info)
    print("EURCAD: currency_base =",symbol_info.currency_base,"  currency_profit =",sy
    print()

 # récupère les propriétés des symboles sous la forme d'un dictionnaire
    print("Affiche symbol_info()._asdict() :")
    symbol_info_dict = symbol_info._asdict()
```

© 2000-2024, MetaQuotes Ltd.

```
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
    print()

   # convertit le dictionnaire en DataFrame et l'imprime
    df=pd.DataFrame(list(symbol_info_dict.items()),columns=['property','value'])
    print("symbol_info_dict() sous la forme de dataframe :")
    print(df)

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()


Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0, ses
EURCAD: currency_base = EUR   currency_profit = CAD   currency_margin = EUR

Affiche symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585217595
  digits=5
  spread=39
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
```

© 2000-2024, MetaQuotes Ltd.

```
option_mode=0
option_right=0
bid=1.55192
bidhigh=1.55842
bidlow=1.5419800000000001
ask=1.5523099999999999
askhigh=1.55915
asklow=1.5436299999999998
last=0.0
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=1e-05
trade_tick_value=0.7043642408362214
trade_tick_value_profit=0.7043642408362214
trade_tick_value_loss=0.7044535553770941
trade_tick_size=1e-05
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-1.1
swap_short=-0.9
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
```

© 2000-2024, MetaQuotes Ltd.

```
  price_greeks_gamma=0.0
  price_greeks_vega=0.0
  price_greeks_rho=0.0
  price_greeks_omega=0.0
  price_sensitivity=0.0
  basis=
  category=
  currency_base=EUR
  currency_profit=CAD
  currency_margin=EUR
  bank=
  description=Euro vs Canadian Dollar
  exchange=
  formula=
  isin=
  name=EURCAD
  page=http://www.google.com/finance?q=EURCAD
  path=Forex\EURCAD

symbol_info_dict() sous la forme de dataframe :
        propriété                                  valeur
0          custom                                   False
1      chart_mode                                       0
2          select                                    True
3         visible                                    True
4   session_deals                                       0
..            ...                                     ...
91        formula
92           isin
93           name                                  EURCAD
94           page  http://www.google.com/finance?q=EURCAD
95           path                             Forex\EURCAD

[96 lignes x 2 colonnes]
```

Voir aussi

symbol_info

© 2000-2024, MetaQuotes Ltd.

market_book_add

Abonne le terminal MetaTrader 5 aux événements de changements du Market Depth pour un symbole spécifié.

```
market_book_add(
   symbol      // nom de l'instrument financier
)
```

symbol

[in] Nom de l'instrument financier. Paramètre sans nom obligatoire.

Valeur de Retour

True en cas de succès, False sinon.

Note

La fonction est similaire à MarketBookAdd.

Voir aussi

market_book_get, market_book_release, Structure du Market Depth

© 2000-2024, MetaQuotes Ltd.

market_book_get

Renvoie un tuple de BookInfo contenant des entrées du Market Depth pour le symbole spécifié.

```
market_book_get(
   symbol      // nom de l'instrument financier
)
```

symbol

[in] Nom de l'instrument financier. Paramètre sans nom obligatoire.

Valeur de Retour

Renvoie le contenu du Market Depth sous forme de tuple à partir des entrées BookInfo avec le type de l'ordre, le prix et le volume en lots. BookInfo est similaire à la structure MqlBookInfo.

Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

L'abonnement aux événements de changement du Market Depth doit être effectué au préalable à l'aide de la fonction market_book_add().

La fonction est similaire à MarketBookGet.

Exemple :

```
import MetaTrader5 as mt5
import time
# afficher les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)
print("")

# établissement de la connexion au terminal MetaTrader 5
if not mt5.initialize():
    print("échec de initialize(), code d'erreur =",mt5.last_error())
   # ferme la connexion au terminal MetaTrader 5
    mt5.shutdown()
    quit()

# abonnement aux mises à jour du Market Depth pour l'EURUSD (Depth of Market)
if mt5.market_book_add('EURUSD'):
  # obtient les données du Market Depth 10 fois en boucle
   for i in range(10):
# obtient le contenu du Market Depth
        items = mt5.market_book_get('EURUSD')
 # affiche l'intégralité du market depth "tel quel" dans une seule chaîne
        print(items)
 # affiche maintenant chaque ordre séparément pour plus de clarté
```

© 2000-2024, MetaQuotes Ltd.

```
        if items:
            for it in items:
 # contenu de l'ordre
                print(it._asdict())
 # pause pendant 5 secondes avant la prochaine demande des données de profondeur du ma
        time.sleep(5)
  # annule l'abonnement aux mises à jour de la profondeur du marché (Depth of Market)
   mt5.market_book_release('EURUSD')
else:
    print("échec de mt5.market_book_add('EURUSD'), code d'erreur =",mt5.last_error())

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

```
Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.34

(BookInfo(type=1, price=1.20038, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20038, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20032, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20023, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20017, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0), BookInfo(ty
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0), BookInfo(ty
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20036, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20036, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
```

© 2000-2024, MetaQuotes Ltd.

```
{'type': 1, 'price': 1.20028, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20035, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20035, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20027, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20037, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20037, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20031, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20022, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20016, 'volume': 250, 'volume_dbl': 250.0}
```

Voir aussi

market_book_add, market_book_release, Structure du Market Depth

© 2000-2024, MetaQuotes Ltd.

market_book_release

Annule l'abonnement du terminal MetaTrader 5 aux événements de changement de Market Depth pour un symbole spécifié.

```
market_book_release(
   symbol      // nom de l'instrument financier
)
```

symbol

[in] Nom de l'instrument financier. Paramètre sans nom obligatoire.

Valeur de Retour

True en cas de succès, False sinon.

Note

La fonction est similaire à MarketBookRelease.

Voir aussi

market_book_add, market_book_get, Structure du Market Depth

© 2000-2024, MetaQuotes Ltd.

copy_rates_from

Retourne les barres du terminal MetaTrader 5 à partir de la date spécifiée.

```
copy_rates_from(
   symbol,          // nom du symbole
   timeframe,       // période
   date_from,       // date de début
   count            // nombre de barres
   )
```

Paramètres

symbol

[in] Nom de l'instrument financier, par exemple, "EURUSD". Paramètre non nommé requis.

timeframe

[in] Période pour laquelle les barres sont demandées. Défini par une valeur de l'énumération TIMEFRAME. Paramètre sans nom requis.

date_from

[in] Date d'ouverture de la première barre de l'échantillon demandé. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Paramètre non nommé requis.

count

[in] Nombre de barres à récupérer. Paramètre non nommé requis.

Valeur de Retour

Renvoie les barres sous forme de tableau numpy avec les colonnes nommées time, open, high, low, close, tick_volume, spread et real_volume. Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

Voir la fonction CopyRates() pour plus d'informations.

Seules les données dont la date est inférieure (donnée plus récente) ou égale à la date spécifiée seront retournées. Cela signifie que l'heure d'ouverture de n'importe quelle barre est toujours inférieure ou égale à celle spécifiée.

Le terminal MetaTrader 5 fournit des barres uniquement dans l'historique disponible pour un utilisateur sur les graphiques. Le nombre de barres disponibles pour les utilisateurs est défini dans le paramètre "Max. Max. barres dans le graphique".

Lors de la création de l'objet 'datetime', Python utilise le fuseau horaire local, tandis que MetaTrader 5 stocke l'heure d'ouverture des ticks et des barres dans le fuseau horaire UTC (sans décalage). Par conséquent, 'datetime' doit être créé en temps UTC pour exécuter des fonctions qui utilisent le temps. Les données reçues du terminal MetaTrader 5 sont en heure UTC.

TIMEFRAME est une énumération avec les valeurs possibles de période du graphique

© 2000-2024, MetaQuotes Ltd.

ID	Description
TIMEFRAME_M1	1 minute
TIMEFRAME_M2	2 minutes
TIMEFRAME_M3	3 minutes
TIMEFRAME_M4	4 minutes
TIMEFRAME_M5	5 minutes
TIMEFRAME_M6	6 minutes
TIMEFRAME_M10	10 minutes
TIMEFRAME_M12	12 minutes
TIMEFRAME_M12	15 minutes
TIMEFRAME_M20	20 minutes
TIMEFRAME_M30	30 minutes
TIMEFRAME_H1	1 heure
TIMEFRAME_H2	2 heures
TIMEFRAME_H3	3 heures
TIMEFRAME_H4	4 heures
TIMEFRAME_H6	6 heures
TIMEFRAME_H8	8 heures
TIMEFRAME_H12	12 heures
TIMEFRAME_D1	1 jour
TIMEFRAME_W1	1 semaine
TIMEFRAME_MN1	1 mois

Exemple :

```
from datetime import datetime
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# importe le module 'pandas' pour afficher les données obtenues sous forme tabulaire
import pandas as pd
pd.set_option('display.max_columns', 500) # nombre de colonnes à afficher
pd.set_option('display.width', 1500)      # largeur max d'affichage de la table
# importe le module pytz pour travailler avec le fuseau horaire
```

© 2000-2024, MetaQuotes Ltd.

```
import pytz

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# définit le fuseau horaire sur UTC
timezone = pytz.timezone("Etc/UTC")
# crée un objet 'datetime' dans le fuseau horaire UTC pour éviter l'implémentation d'u
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# récupère 10 barres H4 de l'EURUSD à partir du 01.10.2020 dans le fuseau horaire UTC
rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_H4, utc_from, 10)

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
# affiche chaque élément des données obtenues dans une nouvelle ligne
print("Affiche les données brutes obtenues")
for rate in rates:
    print(rate)

# crée le DataFrame à partir des données obtenues
rates_frame = pd.DataFrame(rates)
# convertit le temps en secondes au format datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# affiche les données
print("\nAffiche les données sous forme de tableau")
print(rates_frame)
```

```
Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

Affiche les données brutes obtenues
(1578484800, 1.11382, 1.11385, 1.1111, 1.11199, 9354, 1, 0)
(1578499200, 1.11199, 1.11308, 1.11086, 1.11179, 10641, 1, 0)
(1578513600, 1.11178, 1.11178, 1.11016, 1.11053, 4806, 1, 0)
(1578528000, 1.11053, 1.11193, 1.11033, 1.11173, 3480, 1, 0)
(1578542400, 1.11173, 1.11189, 1.11126, 1.11182, 2236, 1, 0)
(1578556800, 1.11181, 1.11203, 1.10983, 1.10993, 7984, 1, 0)
(1578571200, 1.10994, 1.11173, 1.10965, 1.11148, 7406, 1, 0)
(1578585600, 1.11149, 1.11149, 1.10923, 1.11046, 7468, 1, 0)
(1578600000, 1.11046, 1.11097, 1.11033, 1.11051, 3450, 1, 0)
(1578614400, 1.11051, 1.11093, 1.11017, 1.11041, 2448, 1, 0)

Afficher les données sous forme de tableau
                 time     open     high      low    close  tick_volume  spread  real_v
```

© 2000-2024, MetaQuotes Ltd.

```
0 2020-01-08 12:00:00  1.11382  1.11385  1.11110  1.11199         9354       1
1 2020-01-08 16:00:00  1.11199  1.11308  1.11086  1.11179        10641       1
2 2020-01-08 20:00:00  1.11178  1.11178  1.11016  1.11053         4806       1
3 2020-01-09 00:00:00  1.11053  1.11193  1.11033  1.11173         3480       1
4 2020-01-09 04:00:00  1.11173  1.11189  1.11126  1.11182         2236       1
5 2020-01-09 08:00:00  1.11181  1.11203  1.10983  1.10993         7984       1
6 2020-01-09 12:00:00  1.10994  1.11173  1.10965  1.11148         7406       1
7 2020-01-09 16:00:00  1.11149  1.11149  1.10923  1.11046         7468       1
8 2020-01-09 20:00:00  1.11046  1.11097  1.11033  1.11051         3450       1
9 2020-01-10 00:00:00  1.11051  1.11093  1.11017  1.11041         2448       1
```

Voir aussi

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2024, MetaQuotes Ltd.

copy_rates_from_pos

Retourne les barres du terminal MetaTrader 5 à partir de l'index spécifié.

```
copy_rates_from_pos(
   symbol,          // nom du symbole
   timeframe,       // période
   start_pos,       // position de départ
   count            // nombre de barres
   )
```

Paramètres

symbol

[in] Nom de l'instrument financier, par exemple, "EURUSD". Paramètre non nommé requis.

timeframe

[in] Période pour laquelle les barres sont demandées. Défini par une valeur de l'énumération TIMEFRAME. Paramètre sans nom requis.

start_pos

[in] Position de la première barre des données demandées. La numérotation des barres va du présent au passé. Ainsi, la barre zéro signifie la barre actuelle. Paramètre non nommé requis.

count

[in] Nombre de barres à récupérer. Paramètre non nommé requis.

Valeur de Retour

Renvoie les barres sous forme de tableau numpy avec les colonnes nommées time, open, high, low, close, tick_volume, spread et real_volume. Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

Voir la fonction CopyRates() pour plus d'informations.

Le terminal MetaTrader 5 fournit des barres uniquement dans l'historique disponible pour un utilisateur sur les graphiques. Le nombre de barres disponibles pour les utilisateurs est défini dans le paramètre "Max. Max. barres dans le graphique".

Exemple :

```
from datetime import datetime
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# importe le module 'pandas' pour afficher les données obtenues sous forme tabulaire
import pandas as pd
pd.set_option('display.max_columns', 500) # nombre de colonnes à afficher
pd.set_option('display.width', 1500)      # largeur max d'affichage de la table
```

© 2000-2024, MetaQuotes Ltd.

```
# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# récupère 10 barres D1 du GBPUSD de la journée en cours
rates = mt5.copy_rates_from_pos("GBPUSD", mt5.TIMEFRAME_D1, 0, 10)

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
# affiche chaque élément des données obtenues dans une nouvelle ligne
print("Affiche les données brutes obtenues")
for rate in rates:
    print(rate)

# crée le DataFrame à partir des données obtenues
rates_frame = pd.DataFrame(rates)
# convertit le temps en secondes au format datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# affiche les données
print("\nAffiche les données sous forme de tableau")
print(rates_frame)
```

```
Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

Affiche les données brutes obtenues
(1581552000, 1.29568, 1.30692, 1.29441, 1.30412, 68228, 0, 0)
(1581638400, 1.30385, 1.30631, 1.3001, 1.30471, 56498, 0, 0)
(1581897600, 1.30324, 1.30536, 1.29975, 1.30039, 49400, 0, 0)
(1581984000, 1.30039, 1.30486, 1.29705, 1.29952, 62288, 0, 0)
(1582070400, 1.29952, 1.3023, 1.29075, 1.29187, 57909, 0, 0)
(1582156800, 1.29186, 1.29281, 1.28489, 1.28792, 61033, 0, 0)
(1582243200, 1.28802, 1.29805, 1.28746, 1.29566, 66386, 0, 0)
(1582502400, 1.29426, 1.29547, 1.28865, 1.29283, 66933, 0, 0)
(1582588800, 1.2929, 1.30178, 1.29142, 1.30037, 80121, 0, 0)
(1582675200, 1.30036, 1.30078, 1.29136, 1.29374, 49286, 0, 0)

Afficher les données sous forme de tableau
        time     open     high      low    close  tick_volume  spread  real_volume
0 2020-02-13  1.29568  1.30692  1.29441  1.30412        68228       0            0
1 2020-02-14  1.30385  1.30631  1.30010  1.30471        56498       0            0
2 2020-02-17  1.30324  1.30536  1.29975  1.30039        49400       0            0
3 2020-02-18  1.30039  1.30486  1.29705  1.29952        62288       0            0
4 2020-02-19  1.29952  1.30230  1.29075  1.29187        57909       0            0
5 2020-02-20  1.29186  1.29281  1.28489  1.28792        61033       0            0
```

© 2000-2024, MetaQuotes Ltd.

```
6 2020-02-21  1.28802  1.29805  1.28746  1.29566        66386       0            0
7 2020-02-24  1.29426  1.29547  1.28865  1.29283        66933       0            0
8 2020-02-25  1.29290  1.30178  1.29142  1.30037        80121       0            0
9 2020-02-26  1.30036  1.30078  1.29136  1.29374        49286       0            0
```

Voir aussi

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2024, MetaQuotes Ltd.

copy_rates_range

Retourne les barres du terminal MetaTrader 5 dans la plage de dates spécifiée.

```
copy_rates_range(
   symbol,          // nom du symbole
   timeframe,       // période
   date_from,       // date à partir de laquelle les barres sont demandées
   date_to          // date jusqu'à laquelle les barres sont demandées
   )
```

Paramètres

symbol

[in] Nom de l'instrument financier, par exemple, "EURUSD". Paramètre non nommé requis.

timeframe

[in] Période pour laquelle les barres sont demandées. Défini par une valeur de l'énumération TIMEFRAME. Paramètre sans nom requis.

date_from

[in] Date à partir de laquelle les barres sont demandées. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Les barres avec l'heure d'ouverture >= date_from sont retournées. Paramètre non nommé requis.

date_to

[in] Date jusqu'à laquelle les barres sont demandées. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Les barres avec l'heure d'ouverture <= date_to sont renvoyées. Paramètre non nommé requis.

Valeur de Retour

Renvoie les barres sous forme de tableau numpy avec les colonnes nommées time, open, high, low, close, tick_volume, spread et real_volume. Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

Voir la fonction CopyRates() pour plus d'informations.

Le terminal MetaTrader 5 fournit des barres uniquement dans l'historique disponible pour un utilisateur sur les graphiques. Le nombre de barres disponibles pour les utilisateurs est défini dans le paramètre "Max. Max. barres dans le graphique".

Lors de la création de l'objet 'datetime', Python utilise le fuseau horaire local, tandis que MetaTrader 5 stocke l'heure d'ouverture des ticks et des barres dans le fuseau horaire UTC (sans décalage). Par conséquent, 'datetime' doit être créé en temps UTC pour exécuter des fonctions qui utilisent le temps. Les données reçues du terminal MetaTrader 5 sont en heure UTC.

Exemple :

```
from datetime import datetime
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
```

© 2000-2024, MetaQuotes Ltd.

```
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# importe le module 'pandas' pour afficher les données obtenues sous forme tabulaire
import pandas as pd
pd.set_option('display.max_columns', 500) # nombre de colonnes à afficher
pd.set_option('display.width', 1500)      # largeur max d'affichage de la table
# importe le module pytz pour travailler avec le fuseau horaire
import pytz

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize() a échoué, code d'erreur=",mt5.last_error())
    quit()

# définit le fuseau horaire sur UTC
timezone = pytz.timezone("Etc/UTC")
# crée des objets 'datetime' dans le fuseau horaire UTC pour éviter l'implémentation d
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, hour = 13, tzinfo=timezone)
# récupère des barres de USDJPY M5 dans l'intervalle 2020.01.10 00:00 - 2020.01.11 13:
rates = mt5.copy_rates_range("USDJPY", mt5.TIMEFRAME_M5, utc_from, utc_to)

# fermeture de la connexion au terminal MetaTrader 5
mt5.shutdown()

# affiche chaque élément des données obtenues sur une nouvelle ligne
print("Affiche les données brutes obtenues")
counter=0
for rate in rates:
    counter+=1
    if counter<=10:
        print(rate)

# crée le DataFrame à partir des données obtenues
rates_frame = pd.DataFrame(rates)
# convertit le temps en secondes au format 'datetime'
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# affiche les données
print("\nAffiche les données sous forme de tableau")
print(rates_frame.head(10))
```

```
Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

Affiche les données brutes obtenues
(1578614400, 109.513, 109.527, 109.505, 109.521, 43, 2, 0)
(1578614700, 109.521, 109.549, 109.518, 109.543, 215, 8, 0)
(1578615000, 109.543, 109.543, 109.466, 109.505, 98, 10, 0)
(1578615300, 109.504, 109.534, 109.502, 109.517, 155, 8, 0)
(1578615600, 109.517, 109.539, 109.513, 109.527, 71, 4, 0)
(1578615900, 109.526, 109.537, 109.484, 109.52, 106, 9, 0)
(1578616200, 109.52, 109.524, 109.508, 109.51, 205, 7, 0)
```

© 2000-2024, MetaQuotes Ltd.

```
(1578616500, 109.51, 109.51, 109.491, 109.496, 44, 8, 0)
(1578616800, 109.496, 109.509, 109.487, 109.5, 85, 5, 0)
(1578617100, 109.5, 109.504, 109.487, 109.489, 82, 7, 0)

Afficher les données sous forme de tableau
                 time     open     high      low    close  tick_volume  spread  real_v
0 2020-01-10 00:00:00  109.513  109.527  109.505  109.521           43       2
1 2020-01-10 00:05:00  109.521  109.549  109.518  109.543          215       8
2 2020-01-10 00:10:00  109.543  109.543  109.466  109.505           98      10
3 2020-01-10 00:15:00  109.504  109.534  109.502  109.517          155       8
4 2020-01-10 00:20:00  109.517  109.539  109.513  109.527           71       4
5 2020-01-10 00:25:00  109.526  109.537  109.484  109.520          106       9
6 2020-01-10 00:30:00  109.520  109.524  109.508  109.510          205       7
7 2020-01-10 00:35:00  109.510  109.510  109.491  109.496           44       8
8 2020-01-10 00:40:00  109.496  109.509  109.487  109.500           85       5
9 2020-01-10 00:45:00  109.500  109.504  109.487  109.489           82       7
```

Voir aussi

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2024, MetaQuotes Ltd.

copy_ticks_from

Retourne les ticks du terminal MetaTrader 5 à partir de la date spécifiée.

```
copy_ticks_from(
   symbol,          // nom du symbole
   date_from,       // date à laquelle les ticks sont demandés à
   count,           // nombre de ticks demandés
   flags            // combinaison de flags définissant le type de ticks demandés
   )
```

Paramètres

symbol

[in] Nom de l'instrument financier, par exemple, "EURUSD". Paramètre non nommé requis.

from

[in] Date à laquelle les ticks sont demandés. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Paramètre non nommé requis.

count

[in] Nombre de ticks à récupérer. Paramètre non nommé requis.

flags

[in] Un flag pour définir le type des ticks demandés. COPY_TICKS_INFO - ticks avec les modifications Bid et/ou Ask, COPY_TICKS_TRADE - ticks avec les modifications dans Last et Volume, COPY_TICKS_ALL - tous les ticks. Les valeurs des flags sont décrites dans l'énumération COPY_TICKS. Paramètre sans nom requis.

Valeur de Retour

Renvoie des ticks sous forme de tableau numpy avec les colonnes nommées time, bid, ask, last et flags. La valeur 'flags' peut être une combinaison de drapeaux de l'énumération TICK_FLAG. Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

Voir la fonction CopyTicks pour plus d'informations.

Lors de la création de l'objet 'datetime', Python utilise le fuseau horaire local, tandis que MetaTrader 5 stocke l'heure d'ouverture des ticks et des barres dans le fuseau horaire UTC (sans décalage). Par conséquent, 'datetime' doit être créé en temps UTC pour exécuter des fonctions qui utilisent le temps. Les données reçues du terminal MetaTrader 5 sont en heure UTC.

COPY_TICKS définit les types de ticks qui peuvent être demandés en utilisant copy_ticks_from() et copy_ticks_range () fonctions.

ID	Description
COPY_TICKS_ALL	toutes les ticks

© 2000-2024, MetaQuotes Ltd.

ID	Description
COPY_TICKS_INFO	ticks contenant les changements de prix Bid et/ou Ask
COPY_TICKS_TRADE	ticks contenant les changements de prix Last et/ou Volume

TICK_FLAG définit les drapeaux possibles pour les ticks. Ces flags sont utilisés pour décrire les ticks obtenus par les fonctions copy_ticks_from() et copy_ticks_range().

ID	Description
TICK_FLAG_BID	Le prix Bid a changé
TICK_FLAG_ASK	Le prix Ask changé
TICK_FLAG_LAST	Le prix Last a changé
TICK_FLAG_VOLUME	Le Volume a changé
TICK_FLAG_BUY	Le dernier prix Buy a changé
TICK_FLAG_SELL	Le dernier prix Sell a changé

Exemple :

```
from datetime import datetime
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# importe le module 'pandas' pour afficher les données obtenues sous forme tabulaire
import pandas as pd
pd.set_option('display.max_columns', 500) # nombre de colonnes à afficher
pd.set_option('display.width', 1500)      # largeur max d'affichage de la table
# importe le module pytz pour travailler avec le fuseau horaire
import pytz

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
```

© 2000-2024, MetaQuotes Ltd.

```
    quit()

# définit le fuseau horaire sur UTC
timezone = pytz.timezone("Etc/UTC")
# crée un objet 'datetime' dans le fuseau horaire UTC pour éviter l'implémentation d'u
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# demande 100 000 ticks EURUSD à partir du 10.01.2019 dans le fuseau horaire UTC
ticks = mt5.copy_ticks_from("EURUSD", utc_from, 100000, mt5.COPY_TICKS_ALL)
print("Ticks reçus :",len(ticks))

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()

# affiche les données de chaque tick sur une nouvelle ligne
print("Affiche les ticks bruts obtenus")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# crée le DataFrame à partir des données obtenues
ticks_frame = pd.DataFrame(ticks)
# convertit le temps en secondes au format datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# affiche les données
print("\nAffiche le dataframe avec les ticks")
print(ticks_frame.head(10))
```

```
Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

Ticks reçus : 100000
Affiche les ticks bruts obtenus
(1578614400, 1.11051, 1.11069, 0., 0, 1578614400987, 134, 0.)
(1578614402, 1.11049, 1.11067, 0., 0, 1578614402025, 134, 0.)
(1578614404, 1.1105, 1.11066, 0., 0, 1578614404057, 134, 0.)
(1578614404, 1.11049, 1.11067, 0., 0, 1578614404344, 134, 0.)
(1578614412, 1.11052, 1.11064, 0., 0, 1578614412106, 134, 0.)
(1578614418, 1.11039, 1.11051, 0., 0, 1578614418265, 134, 0.)
(1578614418, 1.1104, 1.1105, 0., 0, 1578614418905, 134, 0.)
(1578614419, 1.11039, 1.11051, 0., 0, 1578614419519, 134, 0.)
(1578614456, 1.11037, 1.11065, 0., 0, 1578614456011, 134, 0.)
(1578614456, 1.11039, 1.11051, 0., 0, 1578614456015, 134, 0.)

Affiche le dataframe avec les ticks
                 time      bid      ask  last  volume       time_msc  flags  volume_re
0 2020-01-10 00:00:00  1.11051  1.11069   0.0       0  1578614400987    134          (
1 2020-01-10 00:00:02  1.11049  1.11067   0.0       0  1578614402025    134          (
2 2020-01-10 00:00:04  1.11050  1.11066   0.0       0  1578614404057    134          (
3 2020-01-10 00:00:04  1.11049  1.11067   0.0       0  1578614404344    134          (
```

```
4 2020-01-10 00:00:12  1.11052  1.11064   0.0       0  1578614412106    134          (
5 2020-01-10 00:00:18  1.11039  1.11051   0.0       0  1578614418265    134          (
6 2020-01-10 00:00:18  1.11040  1.11050   0.0       0  1578614418905    134          (
7 2020-01-10 00:00:19  1.11039  1.11051   0.0       0  1578614419519    134          (
8 2020-01-10 00:00:56  1.11037  1.11065   0.0       0  1578614456011    134          (
9 2020-01-10 00:00:56  1.11039  1.11051   0.0       0  1578614456015    134          (
```

Voir aussi

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2024, MetaQuotes Ltd.

copy_ticks_range

Retourne les ticks du terminal MetaTrader 5 pour la plage de dates spécifiée.

```
copy_ticks_range(
   symbol,          // nom du symbole
   date_from,       // date à laquelle les ticks sont demandés
   date_to,         // date jusqu'à laquelle les ticks sont demandés
   flags            // combinaison de flags définissant le type de ticks demandés
   )
```

Paramètres

symbol

[in] Nom de l'instrument financier, par exemple, "EURUSD". Paramètre non nommé requis.

date_from

[in] Date à laquelle les ticks sont demandés. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Paramètre non nommé requis.

date_to

[in] Date jusqu'à laquelle les ticks sont demandés. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Paramètre non nommé requis.

flags

[in] Un flag pour définir le type des ticks demandés. COPY_TICKS_INFO - ticks avec les modifications Bid et/ou Ask, COPY_TICKS_TRADE - ticks avec les modifications dans Last et Volume, COPY_TICKS_ALL - tous les ticks. Les valeurs des flags sont décrites dans l'énumération COPY_TICKS. Paramètre sans nom requis.

Valeur de Retour

Renvoie des ticks sous forme de tableau numpy avec les colonnes nommées time, bid, ask, last et flags. La valeur 'flags' peut être une combinaison de drapeaux de l'énumération TICK_FLAG. Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

Voir la fonction CopyTicks pour plus d'informations.

Lors de la création de l'objet 'datetime', Python utilise le fuseau horaire local, tandis que MetaTrader 5 stocke l'heure d'ouverture des ticks et des barres dans le fuseau horaire UTC (sans décalage). Par conséquent, 'datetime' doit être créé en temps UTC pour exécuter des fonctions qui utilisent le temps. Les données obtenues à partir de MetaTrader 5 ont l'heure UTC, mais Python applique à nouveau le décalage horaire local lors de la tentative d'impression. Ainsi, les données obtenues doivent également être corrigées pour la présentation visuelle.

Exemple :

```
from datetime import datetime
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
```

© 2000-2024, MetaQuotes Ltd.

```
print("Version du package MetaTrader5 : ",mt5.__version__)

# importe le module 'pandas' pour afficher les données obtenues sous forme tabulaire
import pandas as pd
pd.set_option('display.max_columns', 500) # nombre de colonnes à afficher
pd.set_option('display.width', 1500)      # largeur max d'affichage de la table
# importe le module pytz pour travailler avec le fuseau horaire
import pytz

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() a échoué, code d'erreur=",mt5.last_error())
    quit()

# définit le fuseau horaire sur UTC
timezone = pytz.timezone("Etc/UTC")
# crée des objets 'datetime' dans le fuseau horaire UTC pour éviter l'implémentation d
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, tzinfo=timezone)
# demande les ticks AUDUSD entre le 11.01.2020 et le 11.01.2020
ticks = mt5.copy_ticks_range("AUDUSD", utc_from, utc_to, mt5.COPY_TICKS_ALL)
print("Ticks reçus :",len(ticks))

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()

# affiche les données de chaque tick sur une nouvelle ligne
print("Affiche les ticks bruts obtenus")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# crée le DataFrame à partir des données obtenues
ticks_frame = pd.DataFrame(ticks)
# convertit le temps en secondes au format datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# affiche les données
print("\nAffiche le dataframe avec les ticks")
print(ticks_frame.head(10))
```

```
Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

Ticks reçus : 37008
Affiche les ticks bruts obtenus
(1578614400, 0.68577, 0.68594, 0., 0, 1578614400820, 134, 0.)
(1578614401, 0.68578, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614401, 0.68575, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614411, 0.68576, 0.68594, 0., 0, 1578614411388, 130, 0.)
(1578614411, 0.68575, 0.68594, 0., 0, 1578614411560, 130, 0.)
(1578614414, 0.68576, 0.68595, 0., 0, 1578614414973, 134, 0.)
```

© 2000-2024, MetaQuotes Ltd.

```
(1578614430, 0.68576, 0.68594, 0., 0, 1578614430188, 4, 0.)
(1578614450, 0.68576, 0.68595, 0., 0, 1578614450408, 4, 0.)
(1578614450, 0.68576, 0.68594, 0., 0, 1578614450519, 4, 0.)
(1578614456, 0.68575, 0.68594, 0., 0, 1578614456363, 130, 0.)

Affiche le dataframe avec les ticks
                 time      bid      ask  last  volume       time_msc  flags  volume_re
0 2020-01-10 00:00:00  0.68577  0.68594   0.0       0  1578614400820    134          (
1 2020-01-10 00:00:01  0.68578  0.68594   0.0       0  1578614401128    130          (
2 2020-01-10 00:00:01  0.68575  0.68594   0.0       0  1578614401128    130          (
3 2020-01-10 00:00:11  0.68576  0.68594   0.0       0  1578614411388    130          (
4 2020-01-10 00:00:11  0.68575  0.68594   0.0       0  1578614411560    130          (
5 2020-01-10 00:00:14  0.68576  0.68595   0.0       0  1578614414973    134          (
6 2020-01-10 00:00:30  0.68576  0.68594   0.0       0  1578614430188      4          (
7 2020-01-10 00:00:50  0.68576  0.68595   0.0       0  1578614450408      4          (
8 2020-01-10 00:00:50  0.68576  0.68594   0.0       0  1578614450519      4          (
9 2020-01-10 00:00:56  0.68575  0.68594   0.0       0  1578614456363    130          (
```

Voir aussi

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2024, MetaQuotes Ltd.

orders_total

Retourne le nombre d'ordre actifs.

```
orders_total()
```

Valeur de Retour

Valeur entière.

Note

La fonction est similaire à OrdersTotal.

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# vérifie la présence d'ordres actifs
orders=mt5.orders_total()
if orders>0:
    print("Total d'ordres=",orders)
else:
    print("Ordres non trouvés")

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

Voir aussi

orders_get, positions_total

© 2000-2024, MetaQuotes Ltd.

orders_get

Retourne les ordres actifs avec la possibilité de filtrer par symbole ou ticket. Il existe trois options d'appel.

Appel sans paramètres. Retourne les ordres actifs sur tous les symboles.

```
orders_get()
```

Appel spécifiant un symbole pour lequel les ordres actifs sont demandés.

```
orders_get(
   symbol="SYMBOL"       // nom du symbole
)
```

Appel spécifiant un groupe de symboles pour lesquels les ordres actifs sont demandés.

```
orders_get(
   group="GROUP"         // filtre pour sélectionner les ordres pour les symboles
)
```

Appel spécifiant le ticket de l'ordre.

```
orders_get(
   ticket=TICKET         // ticket
)
```

symbol="SYMBOL"

[in] Nom du symbole. Paramètre nommé facultatif. Si un symbole est spécifié, le paramètre *ticket* est ignoré.

group="GROUP"

[in] Le filtre pour organiser un groupe de symboles nécessaires. Paramètre nommé facultatif. Si le groupe est spécifié, la fonction renvoie uniquement les ordres actifs répondant à un critère spécifié pour un nom de symbole.

ticket=TICKET

[in] Ticket de l'ordre (ORDER_TICKET). Paramètre nommé facultatif.

Valeur de Retour

Renvoie les informations sous la forme d'une structure de tuple nommée (namedtuple). Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

La fonction permet de recevoir tous les ordres actifs dans un même appel, comme pour OrdersTotal et OrderSelect.

Le paramètre *group* permet de trier les ordres par symboles. '*' peut être utilisé au début et à la fin d'une chaîne.

Le paramètre *group* peut contenir plusieurs conditions séparées par des virgules. Une condition peut être définie comme un masque à l'aide de '*'. Le symbole de négation logique '!' peut être utilisé pour une exclusion. Toutes les conditions sont appliquées séquentiellement, ce qui signifie que les conditions d'inclusion à un groupe doivent être spécifiées en premier, suivies d'une condition

© 2000-2024, MetaQuotes Ltd.

d'exclusion. Par exemple, *group*="*, !EUR" signifie que les ordres pour tous les symboles doivent être sélectionnés en premier et que ceux contenant "EUR" dans les noms de symboles doivent être exclus par la suite.

Exemple :

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # nombre de colonnes à afficher
pd.set_option('display.width', 1500)      # largeur max d'affichage de la table
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)
print()
# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# affiche les données des ordres actives sur GBPUSD
orders=mt5.orders_get(symbol="GBPUSD")
if orders is None:
    print("Aucun ordre sur GBPUSD, code d'erreur={}".format(mt5.last_error()))
else:
    print("Total d'ordres sur GBPUSD :",len(orders))
    # affiche tous les ordres actifs
    for order in orders:
        print(order)
print()

# récupère la liste des ordres sur les symboles dont les noms contiennent "*GBP*"
gbp_orders=mt5.orders_get(group="*GBP*")
if gbp_orders is None:
    print("Aucun ordre avec group=\"*GBP*\", code d'erreur={}".format(mt5.last_error()
else:
    print("orders_get(group=\"*GBP*\")={}".format(len(gbp_orders)))
 # affiche ces ordres sous forme de tableau à l'aide de pandas.DataFrame
    df=pd.DataFrame(list(gbp_orders),columns=gbp_orders[0]._asdict().keys())
    df.drop(['time_done', 'time_done_msc', 'position_id', 'position_by_id', 'reason',
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    print(df)

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()

Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

Total d'ordres sur GBPUSD: 2
```

© 2000-2024, MetaQuotes Ltd.

```
TradeOrder(ticket=554733548, time_setup=1585153667, time_setup_msc=1585153667718, time
TradeOrder(ticket=554733621, time_setup=1585153671, time_setup_msc=1585153671419, time

orders_get(group="*GBP*")=4
      ticket          time_setup  time_setup_msc  time_expiration  type  type_time  ty
0  554733548 2020-03-25 16:27:47   1585153667718                0     3          0
1  554733621 2020-03-25 16:27:51   1585153671419                0     2          0
2  554746664 2020-03-25 16:38:14   1585154294401                0     3          0
3  554746710 2020-03-25 16:38:17   1585154297022                0     2          0
```

Voir aussi

orders_total, positions_get

© 2000-2024, MetaQuotes Ltd.

order_calc_margin

Retourne la marge dans la devise du compte pour effectuer une opération de trading spécifiée.

```
order_calc_margin(
   action,      // type de l'ordre (ORDER_TYPE_BUY ou ORDER_TYPE_SELL)
   symbol,      // nom du symbole
   volume,      // volume
   price        // prix d'ouverture
   )
```

Paramètres

action

[in] Type de l'ordre dont les valeurs proviennent de l'énumération ORDER_TYPE. Paramètre sans nom requis.

symbol

[in] Nom de l'instrument financier. Paramètre non nommé requis.

volume

[in] Volume de l'opération de trading. Paramètre non nommé requis.

price

[in] Prix d'ouverture. Paramètre non nommé requis.

Valeur de Retour

Valeur réelle en cas de succès, sinon None. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

La fonction permet d'estimer la marge nécessaire pour un type d'ordre spécifié sur le compte courant et dans l'environnement de marché actuel sans tenir compte des ordres en cours et des positions ouvertes. La fonction est similaire à OrderCalcMargin.

ORDER_TYPE

ID	Description
ORDER_TYPE_BUY	Ordre d'achat sur le marché
ORDER_TYPE_SELL	Ordre de vente au marché
ORDER_TYPE_BUY_LIMIT	Achat à cours limité
ORDER_TYPE_SELL_LIMIT	Vente à cours limité
ORDER_TYPE_BUY_STOP	Ordre Buy Stop
ORDER_TYPE_SELL_STOP	Ordre Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	Une fois le prix de l'ordre atteint, l'ordre en attente Buy Limit est placé au prix StopLimit

© 2000-2024, MetaQuotes Ltd.

ID	Description
ORDER_TYPE_SELL_STOP_LIMIT	Une fois le prix de l'ordre atteint, l'ordre en attente Sell Limit est placé au prix StopLimit
ORDER_TYPE_CLOSE_BY	Ordre pour fermer une position par une position opposée

Exemple :

```
import MetaTrader5 as mt5
# affiche les données du package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion au terminal MetaTrader 5
if not mt5.initialize():
 print("initialize() a échoué, code d'erreur=",mt5.last_error())
    quit()

# récupère la devise du compte
account_currency=mt5.account_info().currency
print("Devise du compte :",account_currency)

# arrange la liste des symboles
symbols=("EURUSD","GBPUSD","USDJPY", "USDCHF","EURJPY","GBPJPY")
print("Symboles pour vérifier la marge :", symbols)
action=mt5.ORDER_TYPE_BUY
lot=0.1
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"non trouvé, ignoré")
        continue
    if not symbol_info.visible:
        print(symbol, "n'est pas visible, tente de l'afficher")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}) a échoué, ignoré",symbol)
            continue
    ask=mt5.symbol_info_tick(symbol).ask
    margin=mt5.order_calc_margin(action,symbol,lot,ask)
    if margin != None:
        print("   {} buy {} lots marge : {} {}".format(symbol,lot,margin,account_curre
    else:
        print("order_calc_margin a échoué : , code d'erreur=", mt5.last_error())

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()


Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

Devise du compte : USD

Symboles pour vérifier la marge : ('EURUSD', 'GBPUSD', 'USDJPY', 'USDCHF', 'EURJPY',
```

© 2000-2024, MetaQuotes Ltd.

```
EURUSD buy 0.1 lots marge : 109.91 USD
GBPUSD buy 0.1 lots marge : 122.73 USD
USDJPY buy 0.1 lots marge : 100.0 USD
USDCHF buy 0.1 lots marge : 100.0 USD
EURJPY buy 0.1 lots marge : 109.91 USD
GBPJPY buy 0.1 lots marge : 122.73 USD
```

Voir aussi

order_calc_profit, order_check

© 2000-2024, MetaQuotes Ltd.

order_calc_profit

Retourne le profit dans la devise du compte pour une opération de trading spécifiée.

```
order_calc_profit(
   action,          // type de l'ordre (ORDER_TYPE_BUY ou ORDER_TYPE_SELL)
   symbol,          // nom du symbole
   volume,          // volume
   price_open,      // prix d'ouverture
   price_close      // prix de clôture
   );
```

Paramètres

action

[in] Le type d'ordre peut prendre l'une des deux valeurs de l'énumération ORDER_TYPE : ORDER_TYPE_BUY ou ORDER_TYPE_SELL. Paramètre sans nom requis.

symbol

[in] Nom de l'instrument financier. Paramètre non nommé requis.

volume

[in] Volume de l'opération de trading. Paramètre non nommé requis.

price_open

[in] Prix d'ouverture. Paramètre non nommé requis.

price_close

[in] Prix de clôture. Paramètre non nommé requis.

Valeur de Retour

Valeur réelle en cas de succès, sinon None. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

La fonction permet d'estimer le résultat d'une opération de trading sur le compte courant et dans l'environnement de trading actuel. La fonction est similaire à OrderCalcProfit.

Exemple :

```
import MetaTrader5 as mt5
# affiche les données du package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion au terminal MetaTrader 5
if not mt5.initialize():
 print("initialize() a échoué, code d'erreur=",mt5.last_error())
```

© 2000-2024, MetaQuotes Ltd.

```
    quit()

# récupère la devise du compte
account_currency=mt5.account_info().currency
print("Devise du compte :",account_currency)

# organise la liste des symboles
symbols = ("EURUSD","GBPUSD","USDJPY")
print("Symboles pour vérifier la marge :", symboles)
# estimation du bénéfice d'achat et de vente
lot=1.0
distance=300
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"non trouvé, ignoré")
        continue
    if not symbol_info.visible:
        print(symbol, "n'est pas visible, tente de l'afficher")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}}) a échoué, ignoré",symbol)
            continue
    point=mt5.symbol_info(symbol).point
    symbol_tick=mt5.symbol_info_tick(symbol)
    ask=symbol_tick.ask
    bid=symbol_tick.bid
    buy_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_BUY,symbol,lot,ask,ask+distance*po
    if buy_profit!=None:
        print("   buy {} {} lots : profit de {} points => {} {}".format(symbol,lot,dis
    else:
        print("order_calc_profit(ORDER_TYPE_BUY) a échoué, code d'erreur=",mt5.last_er
    sell_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_SELL,symbol,lot,bid,bid-distance*
    if sell_profit!=None:
        print("   sell {} {} lots : profit de {} points => {} {}".format(symbol,lot,di
    else:
        print("order_calc_profit(ORDER_TYPE_SELL) a échoué, code d'erreur=",mt5.last_e
    print()

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

```
Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

Devise du compte : USD
Symboles pour vérifier la marge : ('EURUSD', 'GBPUSD', 'USDJPY')
  buy EURUSD 1.0 lots : profit de 300 points => 300.0 USD
  sell EURUSD 1.0 lots : profit de 300 points => 300.0 USD

  buy GBPUSD 1.0 lots : profit de 300 points => 300.0 USD
  sell GBPUSD 1.0 lots : profit de 300 points => 300.0 USD
```

© 2000-2024, MetaQuotes Ltd.

```
buy USDJPY 1.0 lots : profit de 300 points => 276.54 USD
sell USDJPY 1.0 lots : profit de 300 points => 278.09 USD
```

Voir aussi

order_calc_margin, order_check

© 2000-2024, MetaQuotes Ltd.

order_check

Vérifie la suffisance des fonds pour effectuer une opération de trading désirée. Les résultats du contrôle sont renvoyés dans une structure MqlTradeCheckResult.

```
order_check(
   request      // structure de la requête
   );
```

Paramètres

request

[in] Structure de type MqlTradeRequest décrivant une action de trading désirée. Paramètre non nommé requis. Un exemple de remplissage d'une requête et le contenu de l'énumération sont décrits ci-dessous.

Valeur de Retour

Le résultat du contrôle en tant que structure MqlTradeCheckResult. Le champ *request* dans la réponse contient la structure de la requête de trading transmise à order_check().

Note

L'envoi réussi d'une requête **n'implique pas que l'opération de trading demandée sera exécutée avec succès**. La fonction order_check est similaire à OrderCheck.

TRADE_REQUEST_ACTIONS

ID	Description
TRADE_ACTION_DEAL	Place un ordre pour une transaction instantanée avec les paramètres spécifiés (définit un ordre de marché)
TRADE_ACTION_PENDING	Place un ordre pour effectuer une transaction à des conditions spécifiées (ordre en attente)
TRADE_ACTION_SLTP	Change le Stop Loss ou le Take Profit d'une position ouverte
TRADE_ACTION_MODIFY	Modifie les paramètres de l'ordre de trading précédemment placé
TRADE_ACTION_REMOVE	Supprime l'ordre en attente précédemment placé
TRADE_ACTION_CLOSE_BY	Ferme une position par une position opposée

ORDER_TYPE_FILLING

ID	Description
ORDER_FILLING_FOK	Cette politique d'exécution signifie qu'un ordre ne peut être exécuté que dans le volume spécifié. Si le montant nécessaire d'un instrument financier n'est actuellement pas disponible sur le marché, l'ordre ne sera pas exécuté. Le volume souhaité peut être composé de plusieurs offres disponibles.

© 2000-2024, MetaQuotes Ltd.

ID	Description
ORDER_FILLING_IOC	Un accord pour exécuter une transaction au volume maximum disponible sur le marché dans le volume spécifié dans l'ordre. Si la demande ne peut pas être entièrement satisfaite, un ordre avec le volume disponible sera exécuté et le volume restant sera annulé.
ORDER_FILLING_RETURN	Cette politique est utilisée uniquement pour les ordres au marché (ORDER_TYPE_BUY et ORDER_TYPE_SELL), les ordres limit et stop limit (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT) et uniquement pour les symboles avec les modes d'exécution Market ou Exchange. S'il est partiellement rempli, un ordre de marché ou à cours limité avec le volume restant n'est pas annulé et sera traité plus tard. Lors de l'activation des ordres ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT, un ordre limite approprié ORDER_TYPE_BUY_LIMIT / ORDER_TYPE_SELL_LIMIT avec le type ORDER_FILLING_RETURN est créé.

ORDER_TYPE_TIME

ID	Description
ORDER_TIME_GTC	L'ordre reste dans la file d'attente jusqu'à ce qu'il soit annulé manuellement
ORDER_TIME_DAY	L'ordre n'est actif que pednant le jour de bourse en cours
ORDER_TIME_SPECIFIED	L'ordre est actif jusqu'à la date spécifiée
ORDER_TIME_SPECIFIED_DAY	L'ordre est actif jusqu'à 23:59:59 du jour spécifié. Si cette heure semble être hors d'une session de trading, l'expiration est traitée à l'heure de trading la plus proche.

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() a échoué, code d'erreur=",mt5.last_error())
    quit()

# récupère la devise du compte
account_currency=mt5.account_info().currency
print("Devise du compte :",account_currency)

# prépare la structure de la requête
symbol="USDJPY"
```

© 2000-2024, MetaQuotes Ltd.

```
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "non trouvé, ne peut pas appeler order_check()")
    mt5.shutdown()
    quit()

# si le symbole n'est pas disponible dans le MarketWatch, il est ajouté
if not symbol_info.visible:
    print(symbol, "n'est pas visible, tentative de l'afficher")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}) a échoué, sortie",symbol)
        mt5.shutdown()
        quit()

# prépare la requête
point=mt5.symbol_info(symbol).point
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": 1.0,
    "type": mt5.ORDER_TYPE_BUY,
    "price": mt5.symbol_info_tick(symbol).ask,
    "sl": mt5.symbol_info_tick(symbol).ask-100*point,
    "tp": mt5.symbol_info_tick(symbol).ask+100*point,
    "deviation": 10,
    "magic": 234000,
    "comment": "python script",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# effectue la vérification et affiche le résultat 'tel quel'
result = mt5.order_check(request)
print(result);
# demande le résultat sous forme de dictionnaire et l'affiche élément par élément
result_dict=result._asdict()
for field in result_dict.keys():
    print("   {}={}".format(field,result_dict[field]))
 # s'il s'agit d'une structure de requête de trading, elle est affichée également élér
    if field=="request":
        traderequest_dict=result_dict[field]._asdict()
        for tradereq_filed in traderequest_dict:
            print("       traderequest: {}={}".format(tradereq_filed,traderequest_dict

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()


Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

Devise du compte : USD
   retcode=0
```

© 2000-2024, MetaQuotes Ltd.

```
    balance=101300.53
    equity=68319.53
    profit=-32981.0
    margin=51193.67
    margin_free=17125.86
    margin_level=133.45308121101692
    comment=Done
    request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY', volume=1.0,
        traderequest: action=1
        traderequest: magic=234000
        traderequest: order=0
        traderequest: symbol=USDJPY
        traderequest: volume=1.0
        traderequest: price=108.081
        traderequest: stoplimit=0.0
        traderequest: sl=107.98100000000001
        traderequest: tp=108.181
        traderequest: deviation=10
        traderequest: type=0
        traderequest: type_filling=2
        traderequest: type_time=0
        traderequest: expiration=0
        traderequest: comment=python script
        traderequest: position=0
        traderequest: position_by=0
```

Voir aussi

order_send, OrderCheck, Types d'opération de trading, Structure de requête de trading, Structure des résultats de vérification d'une requête de trading, Structure de résultat d'une requête de trading

© 2000-2024, MetaQuotes Ltd.

order_send

Envoie une requête au serveur de trading pour effectuer une opération de trading depuis le terminal. La fonction est similaire à OrderSend.

```
order_send(
   request      // structure de la requête
   );
```

Paramètres

request

[in] Structure de type MqlTradeRequest décrivant une action de trading désirée. Paramètre non nommé requis. Un exemple de remplissage d'une requête et le contenu de l'énumération sont décrits ci-dessous.

Valeur de Retour

Résultat d'exécution sous la forme d'une structure MqlTradeResult. Le champ *request* dans la réponse contient la structure de la requête de trading passée à order_send ().

La structure MqlTradeRequest de la requête de trading

Champ	Description
action	Type de l'opération de trading. La valeur peut être l'une des valeurs de l'énumération TRADE_REQUEST_ACTIONS
magic	Identifiant de l'EA. Permet d'organiser le traitement analytique des ordres de trading. Chaque EA peut définir un identifiant unique lors de l'envoi d'une demande de trading
order	Ticket de l'ordre. Requis pour modifier les ordres en attente
symbol	Le nom de l'instrument de trading pour lequel l'ordre est passé. Non requis lors de la modification des ordres et la fermeture des positions
volume	Volume demandé d'une transaction en lots. Un volume réel lors d'une transaction dépend du type d'exécution de l'ordre.
price	Prix auquel un ordre doit être exécuté. Le prix n'est pas fixé en cas d'ordres au marché pour des instruments de type "Exécution au Marché" (SYMBOL_TRADE_EXECUTION_MARKET) de type TRADE_ACTION_DEAL
stoplimit	Un prix auquel un ordre à cours limité est fixé lorsque le prix atteint la valeur 'prix' (cette condition est obligatoire). L'ordre en attente n'est transmis au système de trading qu'à ce moment
sl	Un prix auquel un ordre Stop Loss est activé lorsque le prix évolue dans une direction défavorable
tp	Un prix auquel un ordre Take Profit est activé lorsque le prix évolue dans une direction favorable
deviation	Déviation maximale acceptable par rapport au prix demandé, spécifié en points

© 2000-2024, MetaQuotes Ltd.

Champ	Description
type	Type de l'ordre. La valeur peut être l'une des valeurs de l'énumération ORDER_TYPE
type_filling	Type de remplissage de l'ordre. La valeur peut être l'une des valeurs de ORDER_TYPE_FILLING
type_time	Type d'ordre par expiration. La valeur peut être l'une des valeurs de ORDER_TYPE_TIME
expiration	Délai d'expiration de l'ordre en attente (pour les ordres de type TIME_SPECIFIED)
comment	Commentaire d'un ordre
position	Ticket de la position. Remplissez-le lorsque vous changez et fermez une position pour une identification claire. Habituellement, c'est le même que le ticket de l'ordre qui a ouvert la position.
position_by	Ticket de la position opposée. Il est utilisé lors de la fermeture d'une position par une position opposée (ouverte sur le même symbole mais dans la direction opposée).

Note

Une requête de trading passe plusieurs étapes de vérification sur le serveur de trading. Tout d'abord, la validité de tous les champs *request* nécessaires est vérifiée. S'il n'y a pas d'erreur, le serveur accepte l'ordre pour un traitement ultérieur. Voir la description de la fonction OrderSend pour plus de détails sur l'exécution des opérations de trading.

Exemple :

```
import time
import MetaTrader5 as mt5

# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ", mt5.__author__)
print("Version du package MetaTrader5 : ", mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# prépare la structure de la requête d'achat
symbol = "USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "non trouvé, ne peut pas appeler order_check()")
    mt5.shutdown()
    quit()

# si le symbole n'est pas disponible dans le MarketWatch, il est ajouté
```

© 2000-2024, MetaQuotes Ltd.

```
if not symbol_info.visible:
    print(symbol, "n'est pas visible, tentative de l'afficher")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}}) a échoué, sortie",symbol)
        mt5.shutdown()
        quit()

lot = 0.1
point = mt5.symbol_info(symbol).point
price = mt5.symbol_info_tick(symbol).ask
deviation = 20
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_BUY,
    "price": price,
    "sl": price - 100 * point,
    "tp": price + 100 * point,
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script open",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# envoie une demande de trading
result = mt5.order_send(request)
# vérifie le résultat de l'exécution
print("1. order_send(): buy {} {} lots à {} avec une déviation={} points".format(symbo
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("2. order_send a échoué, retcode={}".format(result.retcode))
   # demande le résultat sous forme de dictionnaire et l'affiche élément par élément
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
 # s'il s'agit de la structure d'une requête de trading, elle est affichée également é
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_
    print("shutdown() et sortie")
    mt5.shutdown()
    quit()

print("2. order_send terminé, ", result)
print("   position ouverte avec POSITION_TICKET={}".format(result.order))
print("   sleep de 2 secondes avant de fermer la position #{}".format(result.order))
time.sleep(2)
```

© 2000-2024, MetaQuotes Ltd.

```
# crée une demande de fermeture
position_id=result.order
price=mt5.symbol_info_tick(symbol).bid
deviation=20
request={
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_SELL,
    "position": position_id,
    "price": price,
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script close",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}
# envoie une demande de trading
result=mt5.order_send(request)
# vérifie le résultat de l'exécution
print("3. ferme la position #{} : sell {} {} lots à {} avec la déviation={} points".fo
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("4. order_send a échoué, retcode={}".format(result.retcode))
    print("   résultat",result)
else:
    print("4. position #{} fermée, {}".format(position_id,result))
   # demande le résultat sous forme de dictionnaire et l'affiche élément par élément
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
 # s'il s'agit de la structure d'une requête de trading, elle est affichée également é
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

```
Résultat
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29
1. order_send(): buy USDJPY 0.1 lots à 108.023 avec une déviation=20 points
2. order_send terminé,  OrderSendResult(retcode=10009, deal=535084512, order=557416535
  position ouverte avec POSITION_TICKET=557416535
  sleep de 2 secondes avant de fermer la position #557416535
3. ferme la position #557416535 : sell USDJPY 0.1 lots à 108.018 avec la déviation=20
4. position #557416535 fermée, OrderSendResult(retcode=10009, deal=535084631, order=55
   retcode=10009
```

© 2000-2024, MetaQuotes Ltd.

```
    deal=535084631
    order=557416654
    volume=0.1
    price=108.015
    bid=108.015
    ask=108.02
    comment=Request executed
    request_id=55
    retcode_external=0
    request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY', volume=0.1,
        traderequest: action=1
        traderequest: magic=234000
        traderequest: order=0
        traderequest: symbol=USDJPY
        traderequest: volume=0.1
        traderequest: price=108.018
        traderequest: stoplimit=0.0
        traderequest: sl=0.0
        traderequest: tp=0.0
        traderequest: deviation=20
        traderequest: type=1
        traderequest: type_filling=2
        traderequest: type_time=0
        traderequest: expiration=0
        traderequest: comment=python script close
        traderequest: position=557416535
        traderequest: position_by=0
```

Voir aussi

order_check, OrderSend,Types d'opération de trading, Structure de requête de trading, Structure des résultats de la vérification d'une requête de trading, Structure de résultat d'une requête de trading

© 2000-2024, MetaQuotes Ltd.

positions_total

Retourne le nombre de positions ouvertes.

```
positions_total()
```

Valeur de Retour

Valeur entière.

Note

La fonction est similaire à PositionsTotal.

Exemple :

```
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# vérifie la présence de positions ouvertes
positions_total=mt5.positions_total()
if positions_total>0:
    print("Total de ositions=",positions_total)
else:
    print("Positions non trouvées")

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

Voir aussi

positions_get, orders_total

© 2000-2024, MetaQuotes Ltd.

positions_get

Retourne les positions ouvertes avec la possibilité de filtrer par symbole ou ticket. Il existe trois options d'appel.

Appel sans paramètres. Renvoie les positions ouvertes pour tous les symboles.

```
positions_get()
```

Appel spécifiant un symbole pour lequel les positions ouvertes sont demandées.

```
positions_get(
   symbol="SYMBOL"      // nom du symbole
)
```

Appel spécifiant un groupe de symboles pour lesquels les positions ouvertes sont demandées.

```
positions_get(
 group="GROUP" // filtre pour sélectionner les positions par symboles
)
```

Appel spécifiant un ticket de position.

```
positions_get(
   ticket=TICKET        // ticket
)
```

Paramètres

symbol="SYMBOL"

[in] Nom du symbole. Paramètre nommé facultatif. Si un symbole est spécifié, le paramètre *ticket* est ignoré.

group="GROUP"

[in] Le filtre pour organiser un groupe de symboles nécessaires. Paramètre nommé facultatif. Si le groupe est spécifié, la fonction renvoie uniquement les positions répondant aux critères spécifiés pour un nom de symbole.

ticket=TICKET

[in] Ticket de la position (POSITION_TICKET). Paramètre nommé facultatif.

Valeur de Retour

Renvoie les informations sous la forme d'une structure de tuple nommée (namedtuple). Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

La fonction permet de recevoir toutes les positions ouvertes dans un appel similaire à PositionsTotal et PositionSelect.

Le paramètre *group* peut contenir plusieurs conditions séparées par des virgules. Une condition peut être définie comme un masque à l'aide de '*'. Le symbole de négation logique '!' peut être utilisé pour une exclusion. Toutes les conditions sont appliquées séquentiellement, ce qui signifie que les conditions d'inclusion à un groupe doivent être spécifiées en premier, suivies d'une condition d'exclusion. Par exemple, *group*= "*, !EUR" signifie que les positions de tous les symboles doivent

© 2000-2024, MetaQuotes Ltd.

être sélectionnées en premier et que celles contenant "EUR" dans les noms de symboles doivent être exclues par la suite.

Exemple :

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # nombre de colonnes à afficher
pd.set_option('display.width', 1500)      # largeur max d'affichage de la table
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)
print()
# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# obtient les positions ouvertes sur USDCHF
positions=mt5.positions_get(symbol="USDCHF")
if positions==None:
    print("Aucune position sur USDCHF, code d'erreur={}".format(mt5.last_error()))
elif len(positions)>0:
    print("Total des positions sur USDCHF =",len(positions))
 # affiche toutes les positions ouvertes
    for position in positions:
        print(position)

# obtient la liste des positions sur les symboles dont les noms contiennent "*USD*"
usd_positions=mt5.positions_get(group="*USD*")
if usd_positions==None:
    print("Aucune position pour group=\"*USD*\", code d'erreur={}".format(mt5.last_er
elif len(usd_positions)>0:
    print("positions_get(group=\"*USD*\")={}".format(len(usd_positions)))
 # affiche ces positions sous forme de tableau à l'aide de pandas.DataFrame
    df=pd.DataFrame(list(usd_positions),columns=usd_positions[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    df.drop(['time_update', 'time_msc', 'time_update_msc', 'external_id'], axis=1, in
    print(df)

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()


Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

positions_get(group="*USD*")=5
```

© 2000-2024, MetaQuotes Ltd.

	ticket	time	type	magic	identifier	reason	volume	price_open
0	548297723	2020-03-18 15:00:55	1	0	548297723	3	0.01	1.09301
1	548655158	2020-03-18 20:31:26	0	0	548655158	3	0.01	1.08676
2	548663803	2020-03-18 20:40:04	0	0	548663803	3	0.01	1.08640
3	548847168	2020-03-19 01:10:05	0	0	548847168	3	0.01	1.09545
4	548847194	2020-03-19 01:10:07	0	0	548847194	3	0.02	1.09536

Voir aussi

positions_total, orders_get

© 2000-2024, MetaQuotes Ltd.

history_orders_total

Retourne le nombre d'ordres dans l'historique de trading dans l'intervalle spécifié.

```
history_orders_total(
   date_from,      // date à partir de laquelle les ordres sont demandées
   date_to,        // date jusqu'à laquelle les ordres sont demandés
   )
```

Paramètres

date_from

[in] Date à partir de laquelle les ordres sont demandés. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Paramètre non nommé requis.

date_to

[in] Date jusqu'à laquelle les ordres sont demandées. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Paramètre non nommé requis.

Valeur de Retour

Valeur entière.

Note

La fonction est similaire à HistoryOrdersTotal.

Exemple :

```
from datetime import datetime
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# récupère le nombre d'ordres dans l'historique
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_total(from_date, datetime.now())
if history_orders>0:
    print("Total des ordres de l'historique=",history_orders)
else:
    print("Aucun ordre trouvé dans l'historique")

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

© 2000-2024, MetaQuotes Ltd.

Voir aussi

history_orders_get, history_deals_total

© 2000-2024, MetaQuotes Ltd.

history_orders_get

Retourne les ordres de l'historique de trading avec la possibilité de filtrer par ticket ou par position. Il existe trois options d'appel.

Appel spécifiant un intervalle de temps. Retourne tous les ordres compris dans l'intervalle spécifié.

```
history_orders_get(
   date_from,           // date à partir de laquelle les ordres sont demandées
   date_to,             // date jusqu'à laquelle les ordres sont demandés
   group="GROUP"        // filtre pour sélectionner les ordres par symboles
   )
```

Appel spécifiant le ticket de l'ordre. Retourne l'ordre avec le ticket spécifié.

```
history_orders_get(
   ticket=TICKET        // ticket de l'ordre
)
```

Appel spécifiant le ticket de la position. Retourne tous les ordres avec le ticket de position spécifié dans la propriété ORDER_POSITION_ID.

```
history_orders_get(
   position=POSITION    // ticket de la position
)
```

Paramètres

date_from

[in] Date à partir de laquelle les ordres sont demandés. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Le paramètre sans nom requis est spécifié en premier.

date_to

[in] Date jusqu'à laquelle les ordres sont demandées. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Le paramètre non nommé requis est spécifié en second.

group="GROUP"

[in] Le filtre pour organiser un groupe de symboles nécessaires. Paramètre nommé facultatif. Si le groupe est spécifié, la fonction renvoie uniquement les ordres répondant à des critères spécifiés pour un nom de symbole.

ticket=TICKET

[in] Ticket de l'ordre qui est demandé. Paramètre facultatif. S'il n'est pas spécifié, le filtre n'est pas appliqué.

position=POSITION

[in] Ticket d'une position (stocké dans ORDER_POSITION_ID) pour laquelle tous les ordres doivent être reçues. Paramètre facultatif. S'il n'est pas spécifié, le filtre n'est pas appliqué.

Valeur de Retour

Renvoie les informations sous la forme d'une structure de tuple nommée (namedtuple). Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

© 2000-2024, MetaQuotes Ltd.

Note

La fonction permet de recevoir tous les ordres d'historique dans une période spécifiée en un seul appel similaire à HistoryOrdersTotal et HistoryOrderSelect.

Le paramètre *group* peut contenir plusieurs conditions séparées par des virgules. Une condition peut être définie comme un masque à l'aide de '*'. Le symbole de négation logique '!' peut être utilisé pour une exclusion. Toutes les conditions sont appliquées séquentiellement, ce qui signifie que les conditions d'inclusion à un groupe doivent être spécifiées en premier, suivies d'une condition d'exclusion. Par exemple, *group*="*, !EUR" signifie que les transactions pour tous les symboles doivent être sélectionnées en premier et que celles contenant "EUR" dans les noms de symboles doivent être exclues par la suite.

Exemple :

```
from datetime import datetime
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # nombre de colonnes à afficher
pd.set_option('display.width', 1500)      # largeur max d'affichage de la table
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)
print()
# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# récupère le nombre d'ordres dans l'historique
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_get(from_date, to_date, group="*GBP*")
if history_orders==None:
    print("Aucun ordre pour le groupe=\"*GBP*\", code d'erreur={}".format(mt5.last_er
elif len(history_orders)>0:
    print("history_orders_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,le
print()

# affiche tous les ordres historiques par un ticket de position
position_id=530218319
position_history_orders=mt5.history_orders_get(position=position_id)
if position_history_orders==None:
    print("Aucun ordre pour la position #{}".format(position_id))
    print("code d'erreur=",mt5.last_error())
elif len(position_history_orders)>0:
    print("Total des ordres de l'historique pour la position #{}: {}".format(position_
   # affiche tous les ordres historiques ayant un ticket de position spécifié
    for position_order in position_history_orders:
        print(position_order)
```

© 2000-2024, MetaQuotes Ltd.

```
    print()
 # affiche ces ordres sous forme de tableau à l'aide de pandas.DataFrame
    df=pd.DataFrame(list(position_history_orders),columns=position_history_orders[0].
    df.drop(['time_expiration','type_time','state','position_by_id','reason','volume_c
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    df['time_done'] = pd.to_datetime(df['time_done'], unit='s')
    print(df)

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()

Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

history_orders_get(2020-01-01 00:00:00, 2020-03-25 17:17:32.058795, group="*GBP*")=14

Total des ordres de l'historique pour la position #530218319: 2
TradeOrder(ticket=530218319, time_setup=1582282114, time_setup_msc=1582282114681, time
TradeOrder(ticket=535548147, time_setup=1583176242, time_setup_msc=1583176242265, time

      ticket          time_setup  time_setup_msc           time_done  time_done_msc  t
0  530218319 2020-02-21 10:48:34   1582282114681 2020-02-21 16:49:37  1582303777582
1  535548147 2020-03-02 19:10:42   1583176242265 2020-03-02 19:10:42  1583176242265
```

Voir aussi

history_deals_total, history_deals_get

© 2000-2024, MetaQuotes Ltd.

history_deals_total

Retourne le nombre de transactions dans l'historique de trading dans l'intervalle spécifié.

```
history_deals_total(
 date_from,      // date à partir de laquelle les transactions sont demandées à
 date_to         // date jusqu'à laquelle les transactions sont demandées
   )
```

Paramètres

date_from

[in] Date à partir de laquelle les transactions sont demandées. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Paramètre non nommé requis.

date_to

[in] Date jusqu'à laquelle les transactions sont demandées. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Paramètre non nommé requis.

Valeur de Retour

Valeur entière.

Note

La fonction est similaire à HistoryDealsTotal.

Exemple :

```
from datetime import datetime
import MetaTrader5 as mt5
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)

# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# obtient le nombre de transactions dans l'historique
from_date=datetime(2020,1,1)
to_date=datetime.now()
deals=mt5.history_deals_total(from_date, to_date)
if deals>0:
    print("Total transactions=",deals)
else:
    print("Transactions non trouvées dans l'historique")

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()
```

© 2000-2024, MetaQuotes Ltd.

Voir aussi

history_deals_get, history_orders_total

© 2000-2024, MetaQuotes Ltd.

history_deals_get

Retourne les transactions de l'historique de trading avec la possibilité de filtrer par ticket ou par position.

Appel spécifiant un intervalle de temps. Retourne toutes les transactions comprises dans l'intervalle spécifié.

```
history_deals_get(
   date_from,          // date à laquelle les transactions sont demandées
   date_to,            // date jusqu'à laquelle les transactions sont demandées
   group="GROUP"       // filtre pour sélectionner les transactions par symboles
   )
```

Appel spécifiant le ticket de l'ordre. Retourne toutes les transactions ayant le ticket d'ordre spécifié dans la propriété DEAL_ORDER.

```
history_deals_get(
   ticket=TICKET        // ticket de l'ordre
)
```

Appel spécifiant le ticket de la position. Retourne toutes les transactions ayant le ticket de position spécifié dans la propriété DEAL_POSITION_ID.

```
history_deals_get(
   position=POSITION    // ticket de la position
)
```

Paramètres

date_from

[in] Date à partir de laquelle les ordres sont demandés. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Le paramètre sans nom requis est spécifié en premier.

date_to

[in] Date jusqu'à laquelle les ordres sont demandées. Défini par l'objet 'datetime' ou en nombre de secondes écoulées depuis 1970.01.01. Le paramètre non nommé requis est spécifié en second.

group="GROUP"

[in] Le filtre pour organiser un groupe de symboles nécessaires. Paramètre nommé facultatif. Si le groupe est spécifié, la fonction renvoie uniquement les transactions répondant à un critère spécifié pour un nom de symbole.

ticket=TICKET

[in] Ticket d'un ordre (stocké dans DEAL_ORDER) pour lequel toutes les transactions sont demandées. Paramètre facultatif. S'il n'est pas spécifié, le filtre n'est pas appliqué.

position=POSITION

[in] Ticket d'une position (stocké dans DEAL_POSITION_ID) pour laquelle toutes les transactions sont demandées. Paramètre facultatif. S'il n'est pas spécifié, le filtre n'est pas appliqué.

Valeur de Retour

© 2000-2024, MetaQuotes Ltd.

Renvoie les informations sous la forme d'une structure de tuple nommée (namedtuple). Retourne None en cas d'erreur. Les informations sur l'erreur peuvent être obtenues en utilisant last_error().

Note

La fonction permet de recevoir toutes les transactions d'historique dans une période spécifiée en un seul appel similaire à HistoryDealsTotal et HistoryDealSelect.

Le paramètre *group* permet de trier les transactions par symboles. '*' peut être utilisé au début et à la fin d'une chaîne.

Le paramètre *group* peut contenir plusieurs conditions séparées par des virgules. Une condition peut être définie comme un masque à l'aide de '*'. Le symbole de négation logique '!' peut être utilisé pour une exclusion. Toutes les conditions sont appliquées séquentiellement, ce qui signifie que les conditions d'inclusion à un groupe doivent être spécifiées en premier, suivies d'une condition d'exclusion. Par exemple, *group*="*, !EUR" signifie que les transactions pour tous les symboles doivent être sélectionnées en premier et que celles contenant "EUR" dans les noms de symboles doivent être exclues par la suite.

Exemple :

```
import MetaTrader5 as mt5
from datetime import datetime
import pandas as pd
pd.set_option('display.max_columns', 500) # nombre de colonnes à afficher
pd.set_option('display.width', 1500)      # largeur max d'affichage de la table
# affiche les données sur le package MetaTrader 5
print("Auteur du package MetaTrader5 : ",mt5.__author__)
print("Version du package MetaTrader5 : ",mt5.__version__)
print()
# établit une connexion avec le terminal MetaTrader 5
if not mt5.initialize():
 print("initialize () a échoué, code d'erreur =",mt5.last_error())
    quit()

# obtient le nombre de transactions dans l'historique
from_date=datetime(2020,1,1)
to_date=datetime.now()
# récupère les transactions des symboles dont les noms contiennent "GBP" dans un inter
deals=mt5.history_deals_get(from_date, to_date, group="*GBP*")
if deals==None:
    print("Aucune transaction pour le groupe=\"*USD*\", code d'erreur={}".format(mt5.l
elif len(deals)> 0:
    print("history_deals_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,ler

# récupère les transactions des symboles dont les noms ne contiennent ni "EUR" ni "GBI
deals = mt5.history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*")
if deals == None:
    print("Aucune transaction, code d'erreur={}".format(mt5.last_error()))
elif len(deals) > 0:
    print("history_deals_get(from_date, to_date, group=\"*,!*EUR*,!*GBP*\") =", len(de
```

© 2000-2024, MetaQuotes Ltd.

```
    # affiche toutes les transactions brutes obtenues
    for deal in deals:
        print("  ",deal)
    print()
 # affiche ces transactions sous forme de tableau à l'aide de pandas.DataFrame
    df=pd.DataFrame(list(deals),columns=deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)
print("")

# récupère toutes les transactions liées à la position #530218319
position_id=530218319
position_deals = mt5.history_deals_get(position=position_id)
if position_deals == None:
    print("Aucune transaction pour la position #{}".format(position_id))
    print("code d'erreur =", mt5.last_error())
elif len(position_deals) > 0:
    print("Transactions de la position #{}: {}".format(position_id, len(position_deals
 # affiche ces transactions sous forme de tableau à l'aide de pandas.DataFrame
    df=pd.DataFrame(list(position_deals),columns=position_deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)

# ferme la connexion au terminal MetaTrader 5
mt5.shutdown()

Résultat :
Auteur du package MetaTrader5 :  MetaQuotes Software Corp.
Version du package MetaTrader5 :  5.0.29

history_deals_get(from_date, to_date, group="*GBP*") = 14

history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*") = 7
   TradeDeal(ticket=506966741, order=0, time=1582202125, time_msc=1582202125419, type=
   TradeDeal(ticket=507962919, order=530218319, time=1582303777, time_msc=158230377758
   TradeDeal(ticket=513149059, order=535548147, time=1583176242, time_msc=158317624226
   TradeDeal(ticket=516943494, order=539349382, time=1583510003, time_msc=158351000389
   TradeDeal(ticket=516943915, order=539349802, time=1583510025, time_msc=158351002505
   TradeDeal(ticket=517139682, order=539557870, time=1583520201, time_msc=158352020122
   TradeDeal(ticket=517139716, order=539557909, time=1583520202, time_msc=158352020297

      ticket      order                time       time_msc  type  entry  magic  positi
0  506966741          0 2020-02-20 12:35:25  1582202125419     2      0      0
1  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0    5302
2  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    5302
3  516943494  539349382 2020-03-06 15:53:23  1583510003895     1      0      0    5393
4  516943915  539349802 2020-03-06 15:53:45  1583510025054     0      0      0    5393
5  517139682  539557870 2020-03-06 18:43:21  1583520201227     0      1      0    5393
6  517139716  539557909 2020-03-06 18:43:22  1583520202971     1      1      0    5393
```

© 2000-2024, MetaQuotes Ltd.

```
Transactions de la position #530218319: 2
      ticket      order                time       time_msc  type  entry  magic  posit
0  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0    530
1  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    530
```

Voir aussi

history_deals_total, history_orders_get

© 2000-2024, MetaQuotes Ltd.

Modèles ONNX en Apprentissage Automatique

ONNX (Open Neural Network Exchange) est un format open-source pour les modèles de machine learning. Ce projet a plusieurs avantages majeurs :

- ONNX est pris en charge par de grandes entreprises telles que Microsoft, Facebook, Amazon et d'autres partenaires.
- Son format ouvert permet les conversions de format entre différentes boîtes à outils d'apprentissage automatique, et Microsoft ONNXMLTools permet de convertir des modèles au format ONNX.
- MQL5 fournit la conversion automatique des types de données pour les entrées et les sorties du modèle si le type de paramètre passé ne correspond pas au modèle.
- Les modèles ONNX peuvent être créés à l'aide de divers outils d'apprentissage automatique. Ils sont actuellement pris en charge dans Caffe2, Microsoft Cognitive Toolkit, MXNet, PyTorch et OpenCV. Des interfaces pour d'autres frameworks et bibliothèques populaires sont également disponibles.
- Avec le langage MQL5, vous pouvez implémenter un modèle ONNX dans une stratégie de trading et l'utiliser avec tous les avantages de la plateforme MetaTrader 5 pour des opérations efficaces dans le marchés financiers.
- Avant de régler un modèle pour le trading en direct, vous pouvez tester le comportement du modèle sur des données historiques dans le testeur de stratégie, sans utiliser d'outils tiers.

MQL5 fournit les fonctions suivantes pour travailler avec ONNX :

Fonction	Action
OnnxCreate	Crée une session ONNX en chargeant un modèle à partir d'un fichier *.onnx
OnnxCreateFromBuffer	Crée une session ONNX en chargeant un modèle à partir d'un tableau de données
OnnxRelease	Ferme une session ONNX
OnnxRun	Exécute un modèle ONNX
OnnxGetInputCount	Retourne le nombre d'entrées dans un modèle ONNX
OnnxGetOutputCount	Retourne le nombre de sorties dans un modèle ONNX
OnnxGetInputName	Retourne le nom de l'entrée d'un modèle par index
OnnxGetOutputName	Retourne le nom de la sortie d'un modèle par index
OnnxGetInputTypeInfo	Retourne la description du type d'entrée à partir du modèle
OnnxGetOutputTypeInfo	Retourne la description du type de sortie du modèle
OnnxSetInputShape	Définit la forme des données d'entrée d'un modèle par index
OnnxSetOutputShape	Définit la forme des données de sortie d'un modèle par index

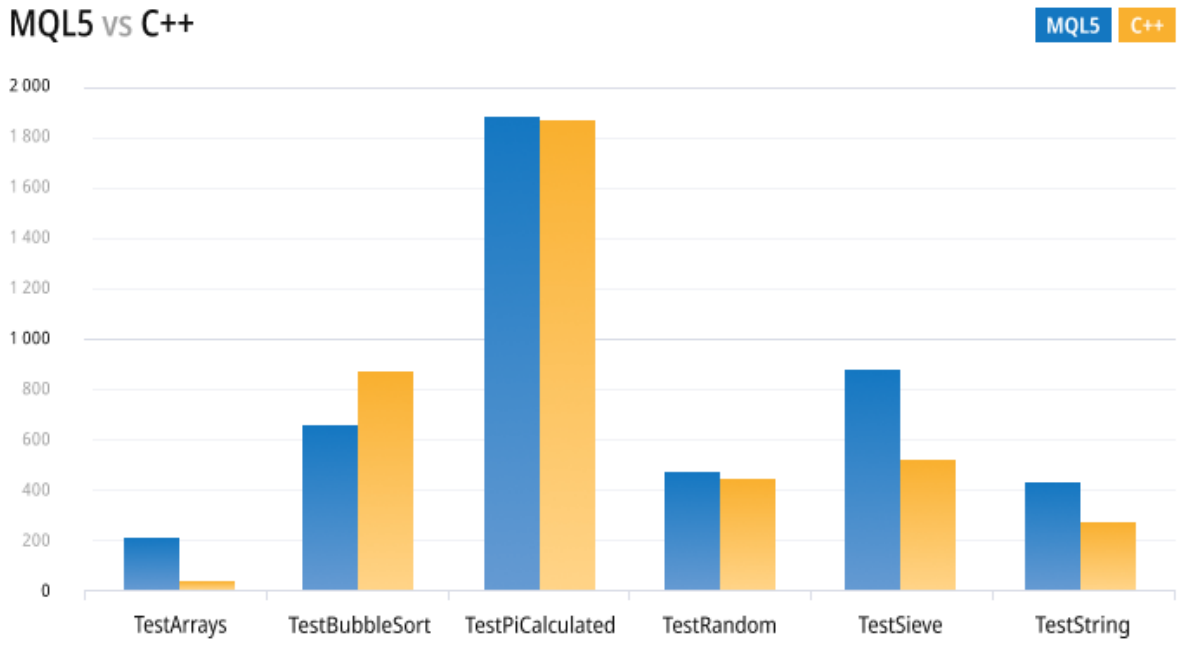
© 2000-2024, MetaQuotes Ltd.

Support ONNX en MQL5

ONNX est une version de format ouvert pour représenter des modèles d'apprentissage automatique. Cette norme définit un ensemble commun d'opérateurs et un format de fichier commun pour permettre aux développeurs d'utiliser des modèles avec différents frameworks, outils, runtimes et compilateurs.

Ainsi, le format ouvert ONNX vous permet de recevoir et de transférer des modèles d'apprentissage automatique entre différentes plateformes et kits d'outils d'apprentissage automatique. ONNX est implémenté dans le langage MQL5 pour permettre aux développeurs d'IA d'exécuter des modèles créés dans l'environnement d'exécution haute performance fourni par la plateforme MetaTrader 5.

La vitesse d'exécution de MQL5 est comparable à celle des applications C++. Ceci est prouvé par les résultats d'exécution des tests standards sur MQL5 et C++. Plus la barre est basse, moins de temps (en millisecondes) est consacré à l'exécution et meilleur est le résultat. Les tests ont été effectués sur Windows 10 (build 17763) x64, Xeon E5-2630 v4 @ 2,20 GHz, mémoire : 65 457 Mo.



Les nouvelles opérations de trading asynchrones et le support natif d'ONNX vous offrent de nouvelles opportunités qui n'étaient auparavant disponibles que pour une poignée de développeurs professionnels d'IA et de traders institutionnels. La prise en charge d'ONNX dans MQL5 permet aux traders de former des modèles pour le trading sur les marchés financiers dans leur environnement de développement préféré, puis de trader avec de faibles coûts de réseau, des vitesses de mise à jour élevées du carnet d'ordres et une soumission asynchrone des ordres.

Actuellement, ONNX est développé et maintenu par des sociétés partenaires telles que Microsoft, Facebook, Amazon et autres, ce qui garantit la poursuite du développement de ce projet ouvert.

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Conversion de Formats

ONNX est un format ouvert qui permet d'utiliser des modèles de différents kits d'outils d'apprentissage automatique. Ce format est pris en charge par de nombreux frameworks, notamment Chainer, Caffee2 et PyTorch.

L'un des outils les plus populaires pour convertir des modèles au format ONNX est ONNXMLToolsde Microsoft.

Les instructions d'installation et d'utilisation d'ONNXMLTools sont disponibles sur le dépôt GitHub. Les kits d'outils suivants sont actuellement pris en charge :

- Keras (un wrapper du convertisseur keras2onnx)
- Tensorflow (un wrapper du convertisseur tf2onnx)
- scikit-learn (un wrapper du convertisseur skl2onnx)
- Apple Core ML
- Spark ML (expérimental)
- LightGBM
- libscm;
- XGBoost;
- H2O
- CatBoost

ONNXMLTools peut être installé facilement. Pour plus de détails sur l'installation et des exemples de conversion de modèle, veuillez consulter la page du projet sur https://github.com/onnx/onnxmltools#install.

© 2000-2024, MetaQuotes Ltd.

Conversion automatique des valeurs d'entrée et de sortie lors de l'exécution de modèles ONNX

La version actuelle d'ONNX dans MQL5 ne prend en charge que les tenseurs pour les valeurs d'entrée/sortie. Les tenseurs sont des tableaux de données avec les éléments des types de données suivants :

Type ONNX	Correspond au type MQL5
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_UINT8	uchar
ONNX_DATA_TYPE_INT8	char
ONNX_DATA_TYPE_UINT16	ushort
ONNX_DATA_TYPE_INT16	short
ONNX_DATA_TYPE_INT32	int
ONNX_DATA_TYPE_INT64	long
ONNX_DATA_TYPE_FLOAT16	—
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	uint
ONNX_DATA_TYPE_UINT64	ulong
ONNX_DATA_TYPE_COMPLEX64	—
ONNX_DATA_TYPE_COMPLEX128	complex
ONNX_DATA_TYPE_BFLOAT16	—
ONNX_DATA_TYPE_STRING	—

Seuls les tableaux, vecteurs et matrices (nous les appellerons **Données**) peuvent être alimentés dans les modèles ONNX en tant que valeurs d'entrée/sortie.

Si les types de paramètres ne correspondent pas au type de paramètre du modèle ONNX et que OnnxRun est appelé sans le flag ONNX_NO_CONVERSION, la conversion automatique des données sera appliquée. L'autoconversion implique qu'avant d'exécuter un modèle ONNX, les **données** de l'utilisateur seront copiées dans les tenseurs ONNX avec la conversion appropriée.

Lorsqu'un modèle ONNX est exécuté sans l'autoconversion, le modèle sera calculé à l'aide des **Données** sans aucune copie supplémentaire.

> **IMPORTANT !** L'autoconversion ne contrôle pas le débordement (troncature), vous devez donc surveiller attentivement les données et les types de données entrés dans le modèle ONNX.

La conversion automatique prend en charge les types ONNX suivants :

© 2000-2024, MetaQuotes Ltd.

- ONNX_DATA_TYPE_BOOL
- ONNX_DATA_TYPE_FLOAT
- ONNX_DATA_TYPE_UINT8
- ONNX_DATA_TYPE_INT8
- ONNX_DATA_TYPE_UINT16
- ONNX_DATA_TYPE_INT16
- ONNX_DATA_TYPE_INT32
- ONNX_DATA_TYPE_INT64
- ONNX_DATA_TYPE_FLOAT16
- ONNX_DATA_TYPE_DOUBLE
- ONNX_DATA_TYPE_UINT32
- ONNX_DATA_TYPE_UINT64
- ONNX_DATA_TYPE_COMPLEX64
- ONNX_DATA_TYPE_COMPLEX128

Types non supportés :

- ONNX_DATA_TYPE_BFLOAT16
- ONNX_DATA_TYPE_STRING

Règles d'Autoconversion par Types de Tensor

Si le type MQL5 n'est pas inclus dans la liste des types pris en charge par le modèle, l'exécution du modèle ONNX renverra l'erreur ERR_ONNX_NOT_SUPPORTED (code d'erreur 5802).

Remarque : lors de la conversion automatique, le type color est traité comme uint, tandis que datetime est traité comme long .

Conversion automatique des valeurs d'entrée

Type ONNX (type d'élément tenseur)	Type MQL5 pris en charge par la conversion automatique
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex Lors de la conversion, les **Données** sont vérifiées par une simple comparaison avec 0
ONNX_DATA_TYPE_FLOAT16	float, double
ONNX_DATA_TYPE_FLOAT	char, uchar, short, ushort, int, color, uint, datetime, long, ulong, float, double
ONNX_DATA_TYPE_UINT8	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	Voir ONNX_DATA_TYPE_FLOAT

© 2000-2024, MetaQuotes Ltd.

Type ONNX (type d'élément tenseur)	Type MQL5 pris en charge par la conversion automatique
ONNX_DATA_TYPE_UINT16	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

Autoconversion des valeurs de sortie

Type ONNX (type d'élément tenseur)	Type MQL5 pris en charge par la conversion automatique
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex Si l'élément tenseur est zéro, alors l'élément Data est mis à 0 ; sinon, la valeur est 1
ONNX_DATA_TYPE_FLOAT16	float, double
ONNX_DATA_TYPE_FLOAT	char, uchar, short, ushort, int, color, uint, datetime, long, ulong, float, double
ONNX_DATA_TYPE_UINT8	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT16	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	Voir ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

© 2000-2024, MetaQuotes Ltd.

Voir également

Conversion de Type

© 2000-2024, MetaQuotes Ltd.

Création d'un modèle

Plusieurs méthodes sont disponibles pour obtenir un modèle prêt au format ONNX. La bibliothèque populaire ONNX Model Zoo contient plusieurs modèles ONNX pré-formés pour différents types de tâches. L'avantage de cette collection est que le bloc-notes de chaque modèle contient des liens vers l'ensemble de données de formation et des références vers l'article original qui décrit l'architecture du modèle.

La plupart des frameworks d'apprentissage automatique utilisent Python. Pour installer le runtime ONNX pour Python, utilisez l'une des commandes suivantes :

```
pip install onnxruntime        # CPU build
pip install onnxruntime-gpu    # GPU build
```

Pour invoquer le runtime ONNX en Python, utilisez la commande suivante :

```
import onnxruntime
session = onnxruntime.InferenceSession("chemin vers le modèle")
```

Pour les entrées et sorties du modèle, consultez la documentation du modèle concerné. Vous pouvez également utiliser des outils de visualisation pour afficher le modèle, tels que Netron ou WinML Dashboard. Dans le runtime ONNX, vous pouvez également interroger les métadonnées d'un modèle et ses entrées et sorties :

```
results = session.run(["output1", "output2"], {
                      "input1": indata1, "input2": indata2})
results = session.run([], {"input1": indata1, "input2": indata2})
```

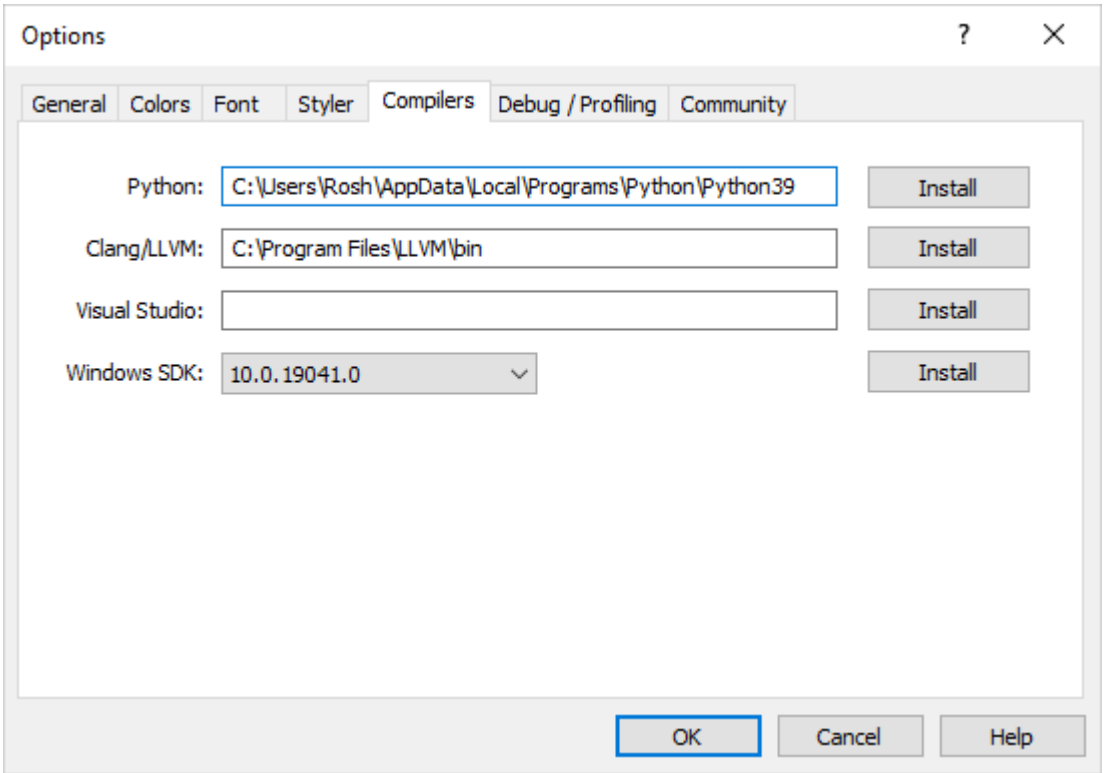
Vous pouvez créer des modèles ONNX directement dans le terminal MetaTrader 5 ou dans MetaEditor en utilisant Python.

Python dans MetaTrader 5

MetaTrader 5 fournit un support prêt à l'emploi pour les scripts Python. Pour activer ces opérations, les développeurs de terminaux fournissent le module MetaTrader5 pour Python : https://pypi.org/project/MetaTrader5.

Dans l'environnement de développement intégré MetaEditor, en plus de créer des applications dans MQL5, vous pouvez également exécuter des scripts Python directement depuis l'éditeur. Pour cela, indiquez le chemin d'accès à l'exécutable dans Paramètres MetaEditor :

© 2000-2024, MetaQuotes Ltd.



Si Python n'est pas installé sur votre ordinateur, cliquez sur Installer pour télécharger le fichier d'installation.

Vous pouvez créer un script Python dans MetaEdtior ou le télécharger dans le dossier de données du terminal et l'exécuter immédiatement à l'aide de la touche F7 (Compiler). Cela ouvrira le terminal MetaTrader 5 avec le script exécuté sur le graphique actuel. Les messages de la console Python (stdout, stderr) seront affichés dans la section Erreurs.

Opérations avec des modèles dans MetaTrader 5

Le langage MQL5 vous permet d'exécuter des modèles ONNX directement dans le terminal MetaTrader 5. Cela se fait en trois étapes :

1. Entraînez le modèle sur une plateforme tierce, telle que Python.
2. Convertissez le modèle en ONNX.
3. Incluez le modèle ONNX dans un Expert Advisor en utilisant la fonction ONNX et exécutez-le dans le terminal MetaTrader 5.

L'intégration de Python dans MQL5 permet d'exécuter un script python et d'enregistrer un modèle ONNX dans MetaEditor ou de l'exécuter directement sur un graphique dans MetaTrader 5. Vous pouvez entraîner le modèle à l'aide d'un script Python pré-écrit aussi souvent que nécessaire directement dans le terminal. La bibliothèque comprend des fonctions prêtes à l'emploi pour obtenir des données de prix, qui peuvent être entrées dans un modèle ONNX :

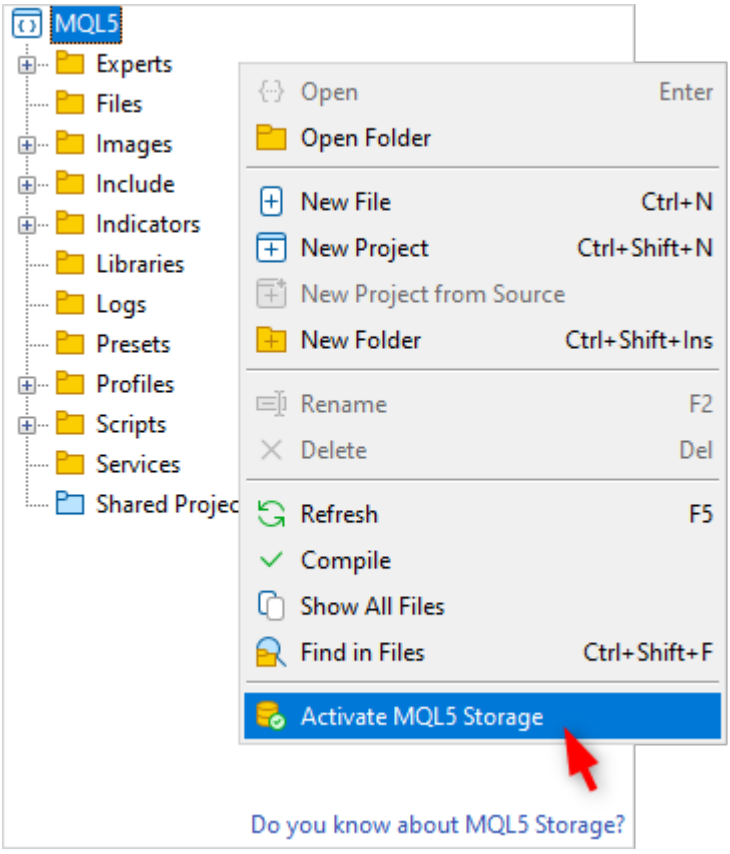
- copy_rates_from - obtient les barres à partir de la date spécifiée
- copy_rates_from_pos - obtient les barres à partir de l'index spécifié
- copy_rates_range - obtient les barres pour l'intervalle de dates spécifié
- copy_ticks_from - obtient les ticks à partir de la date spécifiéee

© 2000-2024, MetaQuotes Ltd.

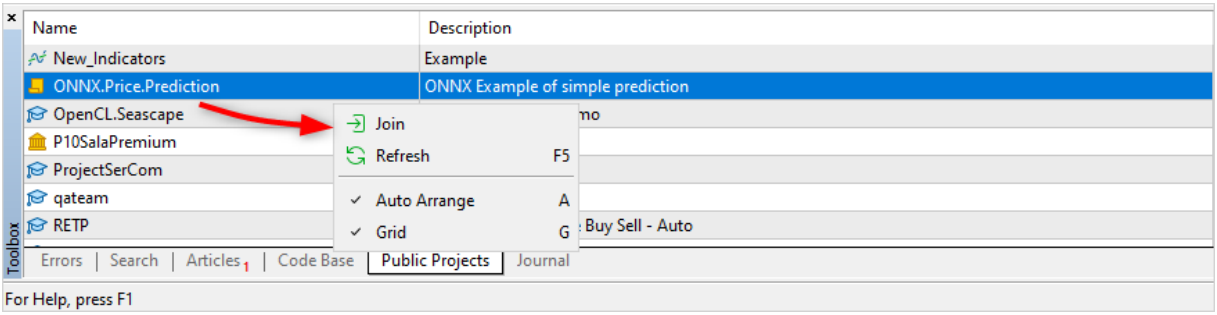
- copy_ticks_range - obtient les ticks pour la plage de dates spécifiée

Exemple de modèle

Un exemple de modèle ONNX fini est disponible dans les projets publics. Vous devez d'abord activer MQL5 Storage dans le Navigateur en spécifiant votre login MQL5 dans les paramètres MetaEditor (sensible à la casse).

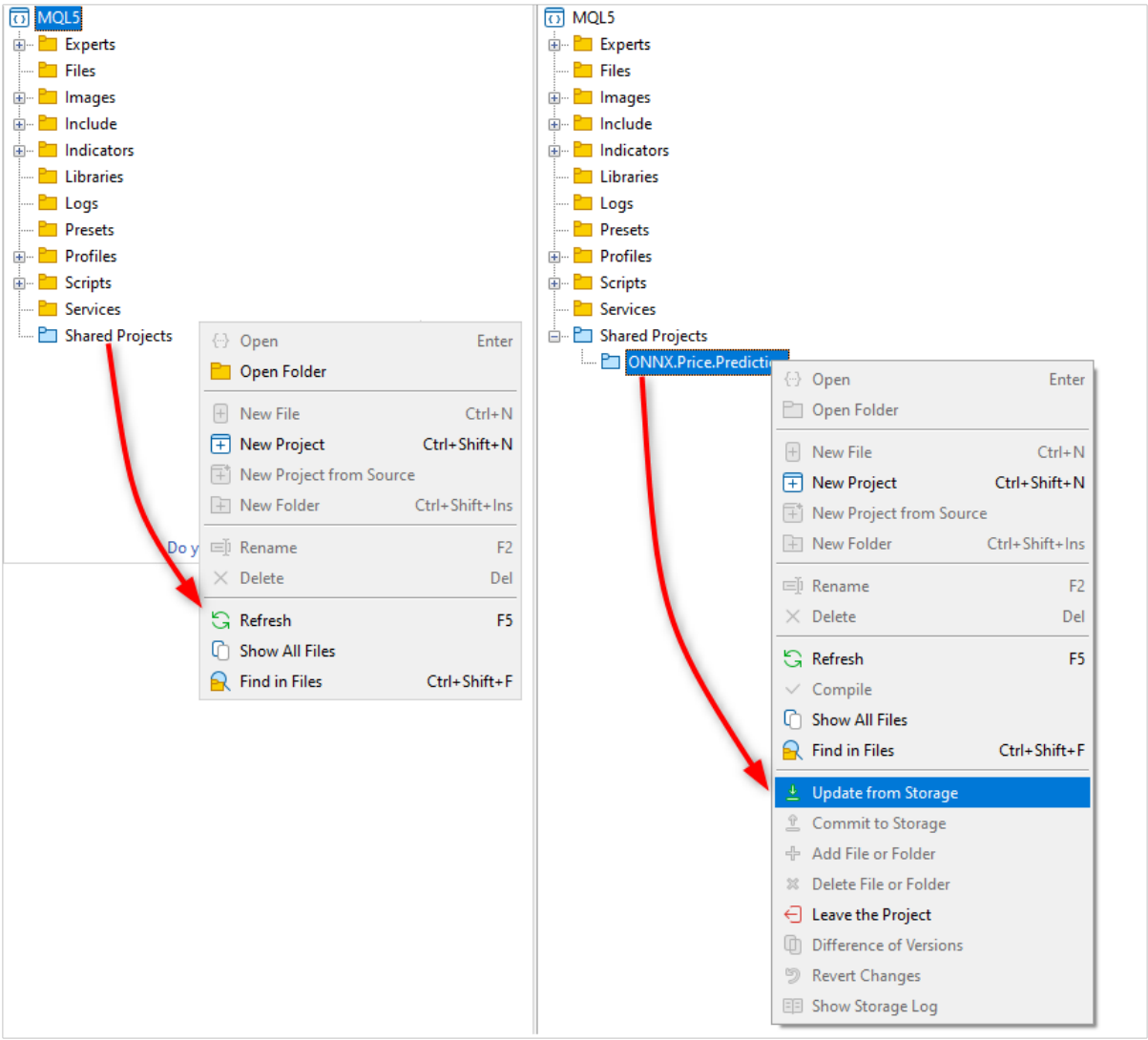


Après l'activation, recherchez le projet ONNX.Price.Prediction et rejoignez-le via la commande du menu contextuel.

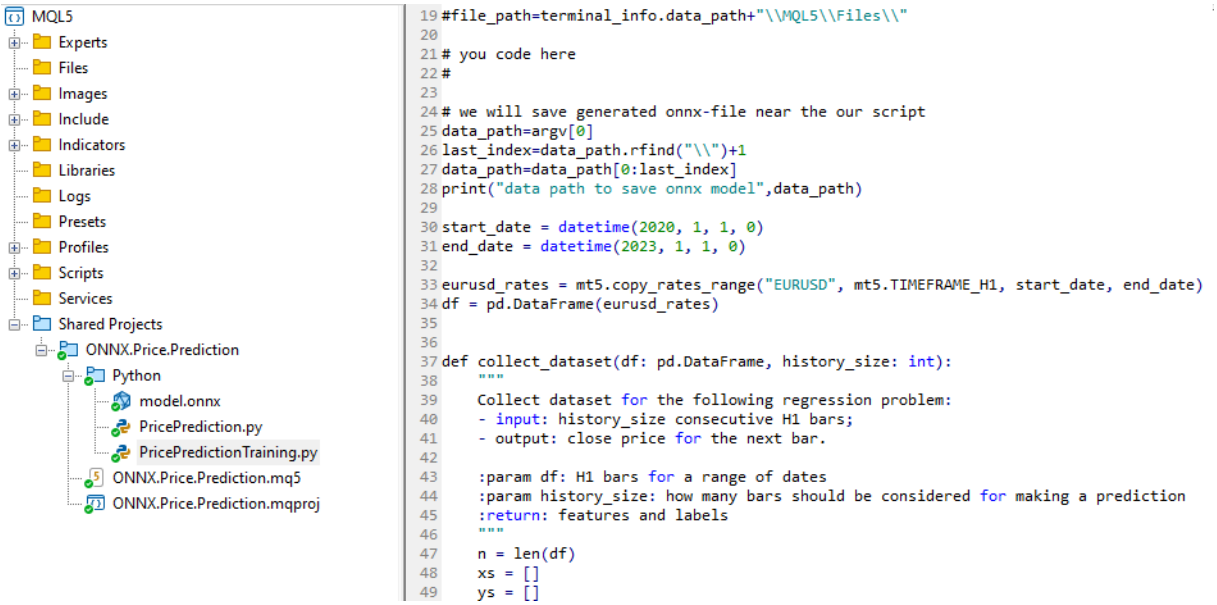


Ensuite, mettez à jour le projet à partir de MQL5 Storage.

© 2000-2024, MetaQuotes Ltd.



Le projet contient un modèle ONNX, deux scripts python, un script MQL5 pour le fonctionnement du projet et un fichier de projet MQL5 (ONNX.Price.Prediction.mqproj).

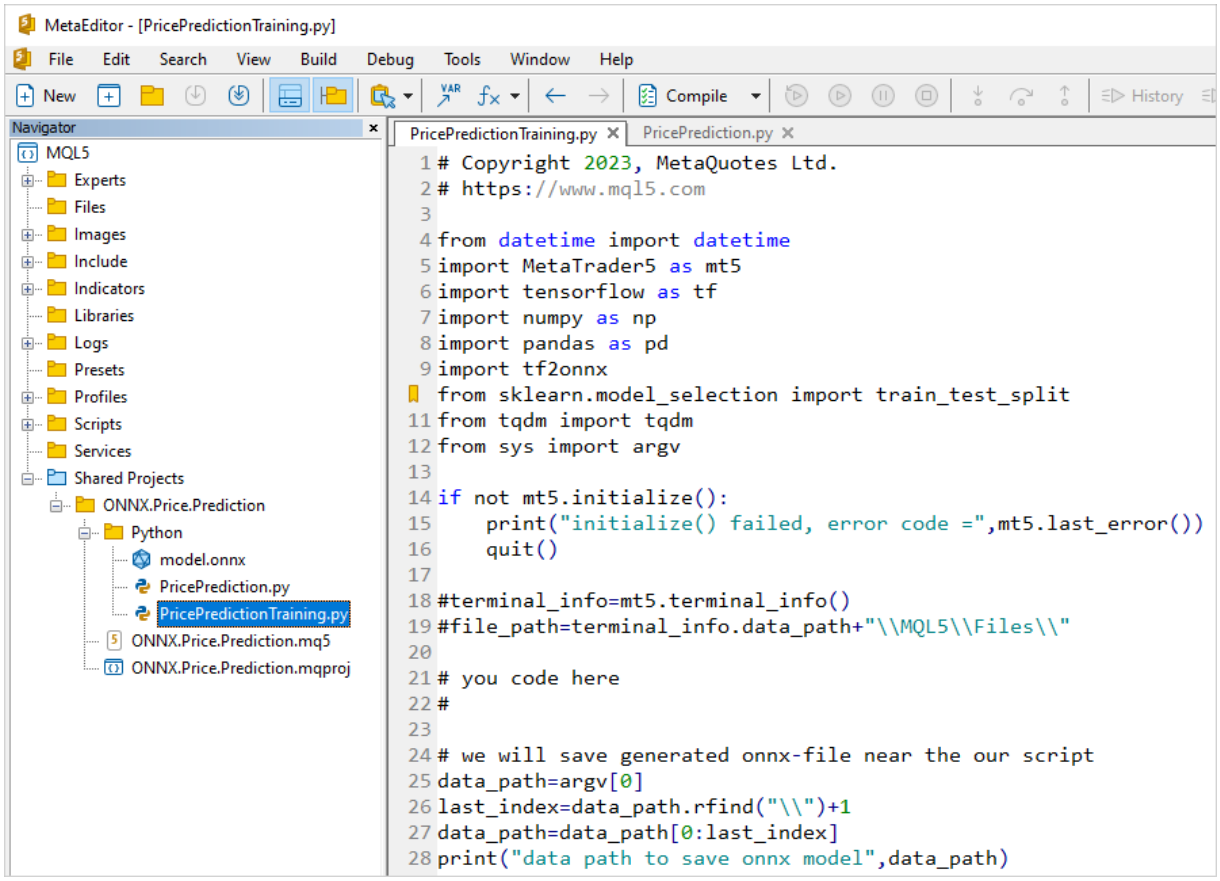


© 2000-2024, MetaQuotes Ltd.

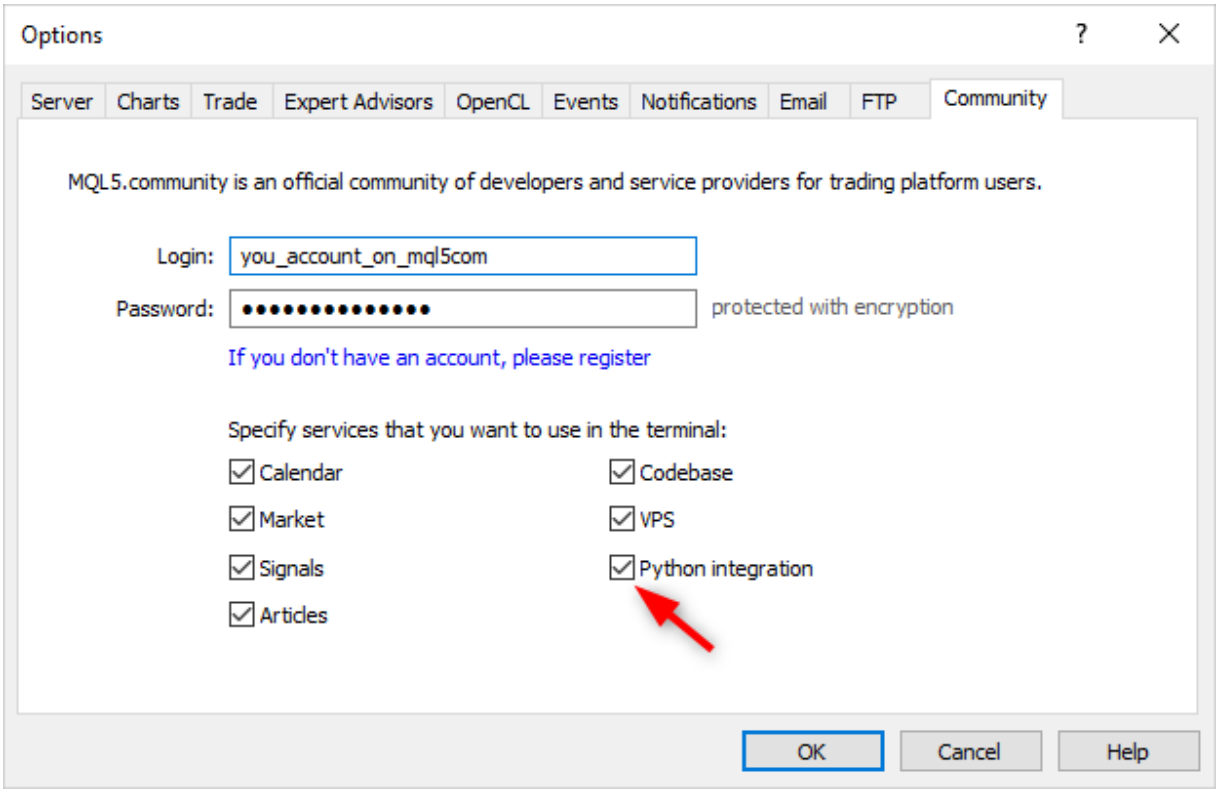
Vous pouvez créer vous-même le modèle ONNX à l'aide du script PricePredictionTraining.py inclus dans le projet. Pour ce faire, vous devez d'abord installer les modules requis à partir de la ligne de commande.

```
python.exe -m pip install --upgrade pip
python -m pip install --upgrade tensorflow
python -m pip install --upgrade pandas
python -m pip install --upgrade scikit-learn
python -m pip install --upgrade matplotlib
python -m pip install --upgrade tqdm
python -m pip install --upgrade metatrader5
python -m pip install --upgrade onnx==1.12
python -m pip install --upgrade tf2onnx
python -m pip install --upgrade numpy
python -m pip install onnxruntime
```

Après avoir installé les modules, ouvrez le script PricePredictionTraining.py dans MetaEditor et exécutez-le avec le bouton Compiler ou avec la touche F7.



Avant d'exécuter le script Python, assurez-vous que le terminal MetaTrader 5 est connecté à un serveur avec le symbole EURUSD disponible. Par exemple, connectez-vous au serveur MetaQuotes-Demo et cochez "Intégration avec Python" dans les paramètresdu terminal.



Lors de la formation du réseau, MetaEditor imprimera les messages du script Python jusqu'à ce que la formation soit terminée.

© 2000-2024, MetaQuotes Ltd.

Description
90%| | 16879/18677 [00:17<00:01, 967.02it/s]
91%| | 16978/18677 [00:17<00:01, 973.79it/s]
91%| | 17079/18677 [00:17<00:01, 981.62it/s]
92%| | 17178/18677 [00:17<00:01, 984.10it/s]
93%| | 17277/18677 [00:17<00:01, 985.85it/s]
93%| | 17377/18677 [00:17<00:01, 987.08it/s]
94%| | 17476/18677 [00:17<00:01, 964.96it/s]
94%| | 17576/18677 [00:18<00:01, 972.40it/s]
95%| | 17676/18677 [00:18<00:01, 980.54it/s]
95%| | 17775/18677 [00:18<00:00, 983.34it/s]
96%| | 17874/18677 [00:18<00:00, 985.32it/s]
96%| | 17973/18677 [00:18<00:00, 980.84it/s]
97%| | 18072/18677 [00:18<00:00, 969.15it/s]
97%| | 18171/18677 [00:18<00:00, 975.30it/s]
98%| | 18271/18677 [00:18<00:00, 979.72it/s]
98%| | 18371/18677 [00:18<00:00, 985.74it/s]
99%| | 18470/18677 [00:18<00:00, 987.00it/s]
99%| | 18569/18677 [00:19<00:00, 973.36it/s]
100%| | 18667/18677 [00:19<00:00, 972.44it/s]
100%| | 18677/18677 [00:19<00:00, 975.79it/s]

Toolbox: Errors | Search | Articles 1 | Code Base | Public Projects | Journal

Lorsque le résultat est de 100 %, le modèle ONNX est prêt et il est enregistré dans le dossier du projet dans <répertoire des données du terminal>\MQL5\Shared Projects\ONNX.Price.Prediction\Python.

Vous pouvez vérifier le modèle résultant en exécutant le deuxième script PricePrediction.py, en appuyant sur **F7**.

Time	Source	Message
2023.03.09 15:11:32.149	Python	[[[1.055292 1.05606 1.054948 1.0556]]]
2023.03.09 15:11:32.149	Python	[[[0.0006845 0.00097058 0.00066865 0.00074513]]]
2023.03.09 15:11:32.149	Python	[[[-1.79986207 -1.24668091 -1.50750979 -1.42256891]
2023.03.09 15:11:32.149	Python	[-1.09861711 -0.99940536 -0.90929162 -0.91259138]
2023.03.09 15:11:32.149	Python	[-0.52885558 -0.74182666 -0.53540526 -0.55023892]
2023.03.09 15:11:32.149	Python	[-0.14901455 -0.52546055 0.07776836 -0.48313661]
2023.03.09 15:11:32.149	Python	[-0.0759682 -0.58727944 -0.16151891 -0.63076169]
2023.03.09 15:11:32.149	Python	[-0.29510726 -0.10303148 0.00299109 0.04026138]
2023.03.09 15:11:32.149	Python	[0.5084026 0.278185 -0.19142981 0.20130692]
2023.03.09 15:11:32.149	Python	[0.66910457 0.94788962 0.15254563 0.33551154]
2023.03.09 15:11:32.149	Python	[0.82980654 0.83455499 0.58625381 1.39572799]
2023.03.09 15:11:32.149	Python	[1.94011106 2.14305478 2.48559649 2.02648968]]]
2023.03.09 15:11:32.181	Python	[[1.9743274]]
2023.03.09 15:11:32.181	Python	predict: [1.05707]

Toolbox: Errors | Search | Articles 1 | Code Base | Public Projects | Journal

For Help, press F1

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Exécution d'un modèle

Pour exécuter un modèle ONNX dans MQL5, effectuez 3 étapes :

1. Chargez le modèle à partir d'un fichier *.onnx à l'aide de la fonction OnnxCreate ou à partir d'un tableau à l'aide de OnnxCreateFromBuffer.
2. Spécifiez les formes de données d'entrée et de sortie à l'aide des fonctions OnnxSetInputShape et OnnxSetOutputShape.
3. Exécutez le modèle à l'aide de la fonction OnnxRun, en lui transmettant les paramètres d'entrée et de sortie pertinents.
4. Si nécessaire, vous pouvez mettre fin à l'opération du modèle à l'aide de la fonction OnnxRelease.

Lors de la création d'un modèle ONNX, vous devez tenir compte des limites et restrictions existantes, qui sont décrites sur https://github.com/microsoft/onnxruntime/blob/rel-1.14.0/docs /OperatorKernels.md

Certains exemples de telles restrictions sont présentés ci-dessous :

Opération	Types de données supportés
ReduceSum	tensor(double), tensor(float), tensor(int32), tensor(int64)
Mul	tensor(bfloat16), tensor(double), tensor(float), tensor(float16), tensor(int32), tensor(int64), tensor(uint32), tensor(uint64)

Vous trouverez ci-dessous un exemple de code MQL5 du projet public ONNX.Price.Prediction.

```
const long   ExtOutputShape[] = {1,1};    // forme de sortie du modèle
const long   ExtInputShape [] = {1,10,4}; // forme d'entrée du modèle
#resource "Python/model.onnx" as uchar ExtModel[]// modèle en tant que ressource
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- récupère 10 barres
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- saisie d'un ensemble de vecteurs OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- remplissage des matrices de normalisation
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
```

© 2000-2024, MetaQuotes Ltd.

```
      ms.Row(s,i);
     }
//--- normalisation des données d'entrée
   x_norm-=mm;
   x_norm/=ms;
//--- crée le modèle
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- spécifie la forme des données d'entrée
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("Echec de OnnxSetInputShape, erreur ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- spécifie la forme des données de sortie
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("Echec de OnnxSetOutputShape, erreur ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- convertit les données d'entrée normalisées en type flottant
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- récupère les données de sortie du modèle ici, c'est-à-dire la prévision de prix
   vectorf y_norm(1);
//--- exécute le modèle
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("Echec de OnnxRun, erreur ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- affiche la valeur de sortie du modèle dans le journal
   Print(y_norm);
//--- effectue la transformation inverse pour obtenir le prix prédit
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("prix prédit :",y_pred);
//--- termine l'opération
   OnnxRelease(handle);
   return(0);
  }
```

Exemple d'exécution du script :

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is tru
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
```

© 2000-2024, MetaQuotes Ltd.

```
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
[0.28188983]
predicted 1.0559258806393044
```

Le terminal MetaTrader 5 a sélectionné l'exécuteur optimal pour les calculs : ONNX Runtime Execution Provider. Dans cet exemple, le modèle a été exécuté sur le CPU.

Modifions le script pour calculer le pourcentage de prédictions de prix de clôture réussies basées sur les valeurs des 10 barres précédentes.

```
#resource "Python/model.onnx" as uchar ExtModel[]// modèle en tant que ressource

#define TESTS 10000  // nombre de jeux de données de test
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- crée le modèle
   long session_handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
   if(session_handle==INVALID_HANDLE)
     {
      Print("Ne peut pas créer le modèle. Erreur ",GetLastError());
      return(-1);
     }

//--- puisque la taille du tenseur d'entrée n'est pas définie pour le modèle, spécifie
//--- le premier index est la taille du lot, le deuxième index est la taille de la sér
   const long input_shape[]={1,10,4};
   if(!OnnxSetInputShape(session_handle,0,input_shape))
     {
      Print("Erreur OnnxSetInputShape ",GetLastError());
      return(-2);
     }

//--- puisque la taille du tenseur de sortie n'est pas définie pour le modèle, spécifi
//--- le premier index est la taille du lot, doit correspondre à la taille du lot dans
//--- le deuxième index est le nombre de prix prédits (seul Close est prédit ici)
   const long output_shape[]={1,1};
   if(!OnnxSetOutputShape(session_handle,0,output_shape))
     {
      Print("Erreur OnnxSetOutputShape ",GetLastError());
      return(-3);
```

© 2000-2024, MetaQuotes Ltd.

```
     }
//--- exécution des tests
   vector closes(TESTS);       // vecteur pour stocker les prix de validation
   vector predicts(TESTS);     // vecteur pour stocker les prédictions obtenues
   vector prev_closes(TESTS); // vecteur pour stocker les prix précédents

   matrix rates;               // matrice pour obtenir la série OHLC
   matrix splitted[2];         // 2 sous-matrices pour diviser la série en test et vali
   ulong  parts[]={10,1};      // tailles des sous-matrices divisées

//--- commence à partir de la barre précédente
   for(int i=1; i<=TESTS; i++)
     {
      //--- récupère 11 barres
      rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,i,11);
      //--- divise la matrice en test et validation
      rates.Vsplit(parts,splitted);
      //--- prend le prix de clôture de la matrice de validation
      closes[i-1]=splitted[1][3][0];
      //--- dernière clôture de la série testée
      prev_closes[i-1]=splitted[0][3][9];

      //--- soumet la matrice de test de 10 barres au test
      predicts[i-1]=PricePredictionTest(session_handle,splitted[0]);
      //--- erreur d'exécution
      if(predicts[i-1]<=0)
        {
         OnnxRelease(session_handle);
         return(-4);
        }
     }
//--- termine l'opération
   OnnxRelease(session_handle);
//--- évalue si le mouvement des prix a été prédit correctement
   int    right_directions=0;
   vector delta_predicts=prev_closes-predicts;
   vector delta_actuals=prev_closes-closes;

   for(int i=0; i<TESTS; i++)
      if((delta_predicts[i]>0 && delta_actuals[i]>0) || (delta_predicts[i]<0 && delta_
         right_directions++;
   PrintFormat("prédictions dans la bonne direction = %.2f%%",(right_directions*100.0)
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| Prépare les données et exécute le modèle                         |
//+------------------------------------------------------------------+
double PricePredictionTest(const long session_handle,matrix& rates)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   static matrixf input_data(10,4); // matrice pour les entrées transformées
   static vectorf output_data(1);   // vecteur pour recevoir le résultat
   static matrix mm(10,4);          // matrice des vecteurs horizontaux Moyenne
   static matrix ms(10,4);          // matrice des vecteurs horizontaux Std

//--- un ensemble de vecteurs verticaux OHLC doit être entré dans le modèle
   matrix x_norm=rates.Transpose();
//--- normalisation des prix
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
   x_norm-=mm;
   x_norm/=ms;

//--- exécute le modèle
   input_data.Assign(x_norm);
   if(!OnnxRun(session_handle,ONNX_DEBUG_LOGS,input_data,output_data))
     {
      Print("Erreur OnnxRun ",GetLastError());
      return(0);
     }
//--- dénormalisation du prix de la valeur de sortie
   double y_pred=output_data[0]*s[3]+m[3];

   return(y_pred);
  }
```

Exécution du script : la précision de la prédiction est d'environ 51 %

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is tru
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
right direction predictions = 51.34 %
```

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Validation du modèle dans le Strategy Tester

Les modèles créés pour les opérations sur les marchés financiers peuvent être validés dans le Strategy Tester du terminal MetaTrader 5. Il s'agit de l'option la plus rapide et la plus pratique, qui élimine le besoin d'émuler en plus l'environnement du marché et les conditions de trading.

Pour tester le modèle, créons un Expert Advisor basé sur le code du projet public ONNX.Price.Prediction. Cela nécessitera quelques modifications.

Déplacez la création du modèle vers la fonction OnInit. La session onnx sera fermée dans OnDeinit. Localisez le bloc d'opérations du modèle principal dans le gestionnaire OnTick.

Ajoutez également l'obtention du prix de clôture des deux barres précédentes, qui est nécessaire pour comparer le prix de clôture réel et la prédiction.

Le code Expert Advisor est petit et facile à lire.

```
const long   ExtInputShape [] = {1,10,4}; // forme d'entrée du modèle
const long   ExtOutputShape[] = {1,1};    // forme de sortie du modèle
#resource "Python/model.onnx" as uchar ExtModel[];// modèle en tant que ressource

long handle;         // handle du modèle
ulong predictions=0; // compteur de prédictions
ulong confirmed=0;   // compteur de prédictions réussies
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'expert                            |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vérifications de base
   if(_Symbol!="EURUSD")
     {
      Print("Le symbole doit être EURUSD, test abandonné");
      return(-1);
     }
   if(_Period!=PERIOD_H1)
     {
      Print("La période doit être H1, test annulé");
      return(-1);
     }
//--- crée le modèle
   handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- spécifie la forme des données d'entrée
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("Echec de OnnxSetInputShape, erreur ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

© 2000-2024, MetaQuotes Ltd.

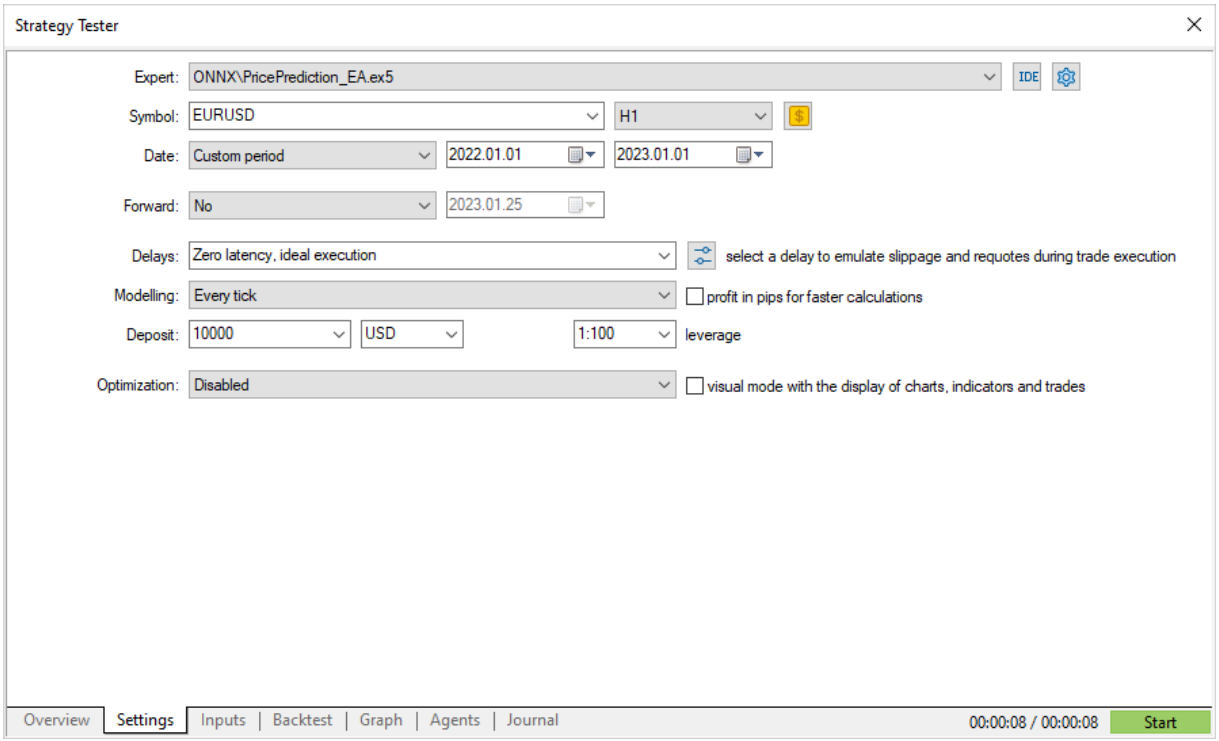
```
//--- spécifie la forme des données de sortie
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("Echec de OnnxSetOutputShape, erreur ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de dé-initialisation de l'expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- termine l'opération du modèle
   OnnxRelease(handle);
//--- calcule et génère des statistiques de prédiction
   PrintFormat("Prédictions réussies = %.2f %%",confirmed*100./double(predictions));
  }
//+------------------------------------------------------------------+
//| Fonction des ticks de l'expert                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
   static datetime open_time=0;
   static double predict;
//--- vérifie l'heure d'ouverture actuelle de la barre
   datetime time=iTime(_Symbol,_Period,0);
   if(time==0)
     {
      PrintFormat("Échec de l'obtention de Time(0), erreur %d", GetLastError());
      return;
     }
//--- si l'heure d'ouverture n'a pas changé, quitter jusqu'au prochain appel OnTick
   if(time==open_time)
      return;
//--- obtient les prix de clôture des deux dernières barres terminées
   double close[];
   int recieved=CopyClose(_Symbol,_Period,1,2,close);
   if(recieved!=2)
     {
      PrintFormat("Echec de CopyClose(2 barres), erreur %d",GetLastError());
      return;
     }
   double delta_predict=predict-close[0]; // changement de prix prévu
   double delta_actual=close[1]-close[0]; // changement de prix réel
   if((delta_predict>0 && delta_actual>0) || (delta_predict<0 && delta_actual<0))
      confirmed++;
```

© 2000-2024, MetaQuotes Ltd.

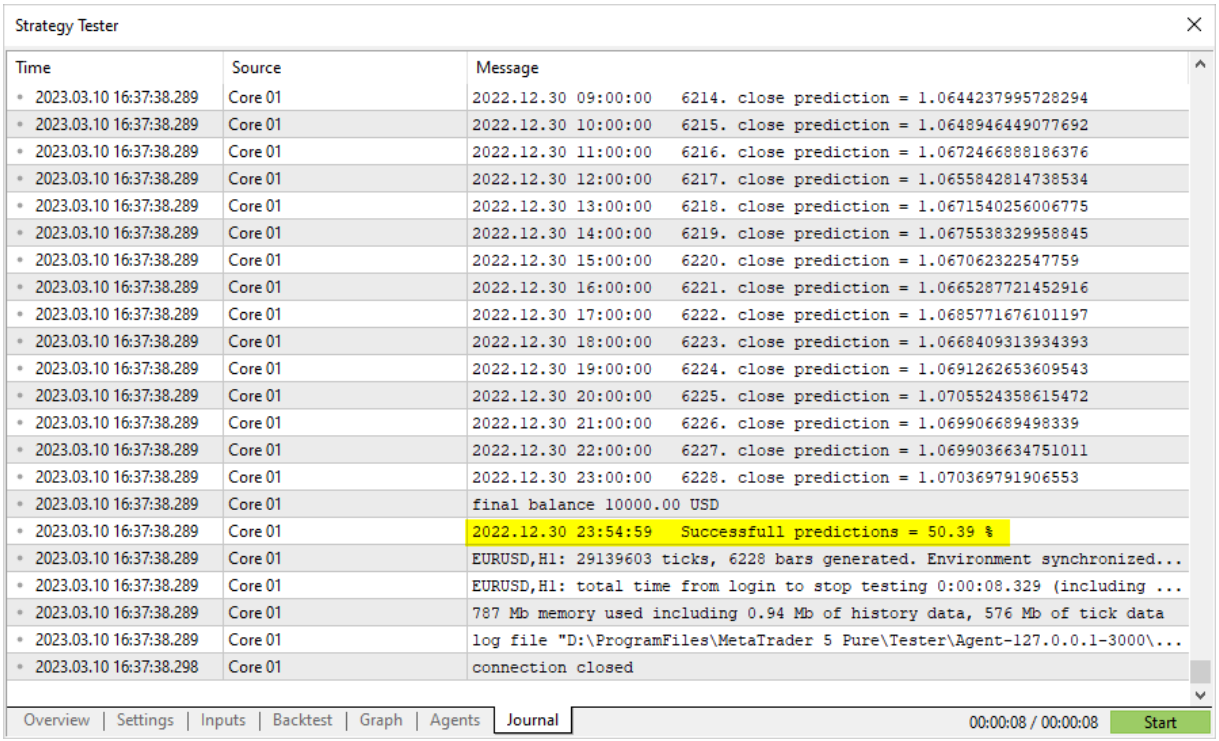
```
//--- calcule le prix de Clôture sur la nouvelle barre pour valider le prix de la bar
   matrix rates;
//--- récupère 10 barres
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,1,10))
      return;
//--- saisie d'un ensemble de vecteurs OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- remplissage des matrices de normalisation
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- normalisation des données d'entrée
   x_norm-=mm;
   x_norm/=ms;
//--- convertit les données d'entrée normalisées en type flottant
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- récupère les données de sortie du modèle ici, c'est-à-dire la prévision de prix
   vectorf y_norm(1);
//--- exécute le modèle
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("Echec de OnnxRun, erreur ",GetLastError());
     }
//--- effectue une transformation inverse pour obtenir le prix prédit et le valider su
   predict=y_norm[0]*s[3]+m[3];
   predictions++;  // augmente le compteur de prédictions
   Print(predictions,". prédiction fermée = ",predict);
//--- enregistre l'heure d'ouverture de la barre pour la vérifier au prochain tick
   open_time=time;
  }
```

Compilez l'Expert Advisor et exécutez les tests pour l'année 2022. Spécifiez EURUSD avec la période H1, qui correspond aux données sur lesquelles le modèle a été formé. Le mode de modélisation des ticks peut être ignoré, puisque le code vérifie l'arrivée d'une nouvelle barre.

© 2000-2024, MetaQuotes Ltd.



Exécutez et vérifiez le résultat dans le journal de test. Il montre qu'un peu plus de 50% des prévisions étaient correctes en 2022.



Strategy Tester

Time	Source	Message
2023.03.10 16:37:38.289	Core 01	2022.12.30 09:00:00 6214. close prediction = 1.0644237995728294
2023.03.10 16:37:38.289	Core 01	2022.12.30 10:00:00 6215. close prediction = 1.0648946449077692
2023.03.10 16:37:38.289	Core 01	2022.12.30 11:00:00 6216. close prediction = 1.0672466888186376
2023.03.10 16:37:38.289	Core 01	2022.12.30 12:00:00 6217. close prediction = 1.0655842814738534
2023.03.10 16:37:38.289	Core 01	2022.12.30 13:00:00 6218. close prediction = 1.0671540256006775
2023.03.10 16:37:38.289	Core 01	2022.12.30 14:00:00 6219. close prediction = 1.0675538329958845
2023.03.10 16:37:38.289	Core 01	2022.12.30 15:00:00 6220. close prediction = 1.067062322547759
2023.03.10 16:37:38.289	Core 01	2022.12.30 16:00:00 6221. close prediction = 1.0665287721452916
2023.03.10 16:37:38.289	Core 01	2022.12.30 17:00:00 6222. close prediction = 1.0685771676101197
2023.03.10 16:37:38.289	Core 01	2022.12.30 18:00:00 6223. close prediction = 1.0668409313934393
2023.03.10 16:37:38.289	Core 01	2022.12.30 19:00:00 6224. close prediction = 1.0691262653609543
2023.03.10 16:37:38.289	Core 01	2022.12.30 20:00:00 6225. close prediction = 1.0705524358615472
2023.03.10 16:37:38.289	Core 01	2022.12.30 21:00:00 6226. close prediction = 1.069906689498339
2023.03.10 16:37:38.289	Core 01	2022.12.30 22:00:00 6227. close prediction = 1.0699036634751011
2023.03.10 16:37:38.289	Core 01	2022.12.30 23:00:00 6228. close prediction = 1.070369791906553
2023.03.10 16:37:38.289	Core 01	final balance 10000.00 USD
2023.03.10 16:37:38.289	Core 01	2022.12.30 23:54:59 Successfull predictions = 50.39 %
2023.03.10 16:37:38.289	Core 01	EURUSD,H1: 29139603 ticks, 6228 bars generated. Environment synchronized...
2023.03.10 16:37:38.289	Core 01	EURUSD,H1: total time from login to stop testing 0:00:08.329 (including ...
2023.03.10 16:37:38.289	Core 01	787 Mb memory used including 0.94 Mb of history data, 576 Mb of tick data
2023.03.10 16:37:38.289	Core 01	log file "D:\ProgramFiles\MetaTrader 5 Pure\Tester\Agent-127.0.0.1-3000\...
2023.03.10 16:37:38.298	Core 01	connection closed

Overview | Settings | Inputs | Backtest | Graph | Agents | Journal 00:00:08 / 00:00:08 Start

Si les tests préliminaires du modèle ont généré des résultats satisfaisants, vous pouvez commencer à écrire une stratégie de trading basée sur ce modèle.

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

OnnxCreate

Crée une session ONNX en chargeant un modèle à partir d'un fichier *.onnx.

```
long  OnnxCreate(
   string  filename,  // chemin vers le fichier
   uint    flags      // flags pour créer le modèle
   );
```

Paramètres

filename

[in] Chemin d'accès au fichier *.onnx du modèle relatif au dossier \MQL5\Files\.

flags

[in] Drapeaux de ENUM_ONNX_FLAGS, décrivant le mode de création du modèle : ONNX_COMMON_FOLDER et ONNX_DEBUG_LOGS.

Valeur de Retour

Le handle de la session créée ou INVALID_HANDLE si une erreur se produit. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

Note

Si le fichier spécifié est introuvable sur le disque, le système réessaye d'ouvrir le fichier, en ajoutant l'extension « .onnx » au nom.

© 2000-2024, MetaQuotes Ltd.

OnnxCreateFromBuffer

Crée une session ONNX, en chargeant un modèle depuis un tableau de données.

```
long  OnnxCreateFromBuffer(
   const uchar&  buffer[],    // référence de tableau
   ulong         flags        // flags de création du modèle
   );
```

Paramètres

buffer

[in] Tableau avec les données du modèle ONNX.

flags

[in] Drapeaux de ENUM_ONNX_FLAGS, décrivant le mode de création du modèle : ONNX_COMMON_FOLDER et ONNX_DEBUG_LOGS.

Valeur de Retour

Le handle de la session créée ou INVALID_HANDLE si une erreur se produit. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxRelease

Ferme une session ONNX.

```
bool  OnnxRelease(
   long   onnx_handle  // handle d'une session ONNX
   );
```

Paramètres

onnx_handle

[in] Descripteur d'objet de session ONNX créé via OnnxCreate ou OnnxCreateFromBuffer.

Valeur de Retour

Renvoie true en cas de succès ; sinon renvoie false. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxRun

Exécute un modèle ONNX.

```
bool  OnnxRun(
   long    onnx_handle,  // handle de session ONNX
   ulong   flags,        // flags décrivant le mode d'exécution
   ...                   // entrées et sorties du modèle
   );
```

Paramètres

onnx_handle

[in] Descripteur d'objet de session ONNX créé via OnnxCreate ou OnnxCreateFromBuffer.

flags

[in] Indicateurs de ENUM_ONNX_FLAGS décrivant le mode d'exécution : ONNX_DEBUG_LOGS et ONNX_NO_CONVERSION.

...

[in] [out] Entrées et sorties du modèle.

Renvoie true en cas de succès ou false sinon. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

ENUM_ONNX_FLAGS

ID	Description
ONNX_DEBUG_LOGS	Journaux de débogage de sortie
ONNX_NO_CONVERSION	Désactive la conversion automatique, utiliser les données utilisateur telles quelles
ONNX_COMMON_FOLDER	Charge un fichier de modèle à partir du dossier Common\Files ; la valeur est égale au drapeau FILE_COMMON

Exemple :

```
const long                              ExtOutputShape[] = {1,1};     // forme de sortie
const long                              ExtInputShape [] = {1,10,4}; // forme d'entrée
#resource "Python/model.onnx" as uchar ExtModel[]                    // modèle en tant
//+------------------------------------------------------------------+
//| Fonction de démarrage du programme                               |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- récupère 10 barres
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- saisie d'un ensemble de vecteurs OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- remplissage des matrices de normalisation
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- normalisation des données d'entrée
   x_norm-=mm;
   x_norm/=ms;
//--- crée le modèle
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- spécifie la forme des données d'entrée
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("Echec de OnnxSetInputShape, erreur ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- spécifie la forme des données de sortie
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("Echec de OnnxSetOutputShape, erreur ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- convertit les données d'entrée normalisées en type flottant
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- récupère les données de sortie du modèle ici, c'est-à-dire la prévision de prix
   vectorf y_norm(1);
//--- exécute le modèle
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("Echec de OnnxRun, erreur ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- affiche la valeur de sortie du modèle dans le journal
   Print(y_norm);
//--- effectue la transformation inverse pour obtenir le prix prédit
   double y_pred=y_norm[0]*s[3]+m[3];
```

© 2000-2024, MetaQuotes Ltd.

```
   Print("prix prédit :",y_pred);
//--- opération terminée
   OnnxRelease(handle);
   return(0);
  };
```

Voir également

OnnxSetInputShape, OnnxSetOutputShape

© 2000-2024, MetaQuotes Ltd.

OnnxGetInputCount

Retourne le nombre d'entrées dans un modèle ONNX.

```
long  OnnxGetInputCount(
   long   onnx_handle  // handle d'une session ONNX
   );
```

Paramètres

onnx_handle

[in] Descripteur d'objet de session ONNX créé via OnnxCreate ou OnnxCreateFromBuffer.

Valeur de Retour

Retourne le nombre de paramètres d'entrée en cas de succès, sinon retourne -1. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxGetOutputCount

Retourne le nombre de sorties dans un modèle ONNX.

```
long  OnnxGetOutputCount(
   long   onnx_handle  // handle d'une session ONNX
   );
```

Paramètres

onnx_handle

[in] Descripteur d'objet de session ONNX créé via OnnxCreate ou OnnxCreateFromBuffer.

Valeur de Retour

Renvoie le nombre de paramètres de sortie en cas de succès ; sinon renvoie -1. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxGetInputName

Retourne le nom de l'entrée d'un modèle par index.

```
string  OnnxGetInputName(
   long   onnx_handle,  // handle de session ONNX
   long   index         // indice du paramètre
   );
```

Paramètres

onnx_handle

[in] Descripteur d'objet de session ONNX créé via OnnxCreate ou OnnxCreateFromBuffer.

index

[in] Index du paramètre d'entrée, commençant par 0.

Valeur de Retour

Retourne le nom du paramètre d'entrée en cas de succès ; sinon renvoie NULL. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxGetOutputName

Retourne le nom de la sortie d'un modèle par index.

```
string  OnnxGetOutputName(
   long   onnx_handle,  // handle de session ONNX
   long   index         // indice du paramètre
   );
```

Paramètres

onnx_handle

[in] Descripteur d'objet de session ONNX créé via OnnxCreate ou OnnxCreateFromBuffer.

index

[in] Index du paramètre de sortie, commençant par 0.

Valeur de Retour

Renvoie le nom du paramètre de sortie en cas de succès ; sinon renvoie NULL. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxGetInputTypeInfo

Retourne la description du type d'entrée à partir du modèle.

```
bool  OnnxGetInputTypeInfo(
   long           onnx_handle,  // handle de session ONNX
   long           index,        // index du paramètre
   OnnxTypeInfo&  typeinfo      // description du type de paramètre
   );
```

Paramètres

onnx_handle

[in] Descripteur d'objet de session ONNX créé via OnnxCreate ou OnnxCreateFromBuffer.

index

[in] Index du paramètre d'entrée, commençant par 0.

typeinfo

[out] La structure OnnxTypeInfo qui décrit le type du paramètre d'entrée.

Valeur de Retour

Renvoie true en cas de succès ; sinon renvoie false. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxGetOutputTypeInfo

Retourne la description du type de sortie à partir du modèle.

```
bool  OnnxGetOutputTypeInfo(
   long           onnx_handle,  // handle de session ONNX
   long           index,        // index du paramètre
   OnnxTypeInfo&  typeinfo      // description du type de paramètre
   );
```

Paramètres

onnx_handle

[in] Descripteur d'objet de session ONNX créé via OnnxCreate ou OnnxCreateFromBuffer.

index

[in] Index du paramètre de sortie, commençant par 0.

typeinfo

[out] La structure OnnxTypeInfo qui décrit le type du paramètre de sortie.

Valeur de Retour

Renvoie true en cas de succès ; sinon renvoie false. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxSetInputShape

Définit la forme des données d'entrée d'un modèle par index.

```
bool  OnnxSetInputShape(
   long          onnx_handle,  // handle de session ONNX
   long          input_index,  // index des paramètres d'entrée
   const ulong&  shape[]       // tableau décrivant la forme des données d'entrée
   );
```

Paramètres

onnx_handle

[in] Descripteur d'objet de session ONNX créé via OnnxCreate ou OnnxCreateFromBuffer.

input_index

[in] Index du paramètre d'entrée, à partir de 0.

shape

[in] Tableau décrivant la forme des données d'entrée du modèle.

Valeur de Retour

Retourne le nom du paramètre d'entrée en cas de succès ; sinon renvoie NULL. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

Exemple :

```
//---- décrit les formes des données d'entrée et de sortie du modèle
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- crée le modèle
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- spécifie la forme des données d'entrée
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("échec, erreur OnnxSetInputShape ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- spécifie la forme des données de sortie
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("échec, erreur OnnxSetOutputShape ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Voir également

OnnxSetOutputShape

© 2000-2024, MetaQuotes Ltd.

OnnxSetOutputShape

Définit la forme des données de sortie d'un modèle par index.

```
bool  OnnxSetOutputShape(
   long          onnx_handle,    // handle de session ONNX
   long          output_index,   // index des paramètres de sortie
   const ulong&  shape[]         // tableau décrivant la forme des données de sortie
   );
```

Paramètres

onnx_handle

[in] Descripteur d'objet de session ONNX créé via OnnxCreate ou OnnxCreateFromBuffer.

output_index

[in] Index du paramètre de sortie, à partir de 0.

shape

[in] Tableau décrivant la forme des données de sortie du modèle.

Valeur de Retour

Retourne le nom du paramètre d'entrée en cas de succès ; sinon renvoie NULL. Pour obtenir le code d'erreur, appelez la fonction GetLastError.

Exemple :

```
//---- décrit les formes des données d'entrée et de sortie du modèle
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- crée le modèle
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- spécifie la forme des données d'entrée
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("échec, erreur OnnxSetInputShape ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- spécifie la forme des données de sortie
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("échec, erreur OnnxSetOutputShape ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Voir également

OnnxSetInputShape

© 2000-2024, MetaQuotes Ltd.

Structures de données

Les structures de données suivantes sont utilisées pour les opérations avec les modèles ONNX :

OnnxTypeInfo

La structure décrit le type d'un paramètre d'entrée ou de sortie d'un modèle ONNX

```
struct OnnxTypeInfo
 {
  ENUM_ONNX_TYPE        type;          // type de paramètre
  OnnxTensorTypeInfo    tensor;        // description du tenseur
  OnnxMapTypeInfo       map;           // description de la carte
  OnnxSequenceTypeInfo  sequence;      // description de la séquence
 };
```

Seul le tenseur (ONNX_TYPE_TENSOR) peut être utilisé comme entrée. Dans ce cas, seul le champ OnnxTypeInfo::tensor est rempli de valeurs, tandis que les autres champs (carte et séquence) ne sont pas définis.

Un seul des trois types OnnxTypeInfo (ONNX_TYPE_TENSOR, ONNX_TYPE_MAP ou ONNX_TYPE_SEQUENCE) peut être utilisé comme entrée. La sous-structure correspondante (OnnxTypeInfo::tensor, OnnxTypeInfo::map ou OnnxTypeInfo::sequence) est remplie en fonction du type.

OnnxTensorTypeInfo

La structure décrit le tenseur dans le paramètre d'entrée ou de sortie d'un modèle ONNX

```
struct OnnxTensorTypeInfo
 {
  const ENUM_ONNX_DATA_TYPE   data_type;     // type de données dans le tenseur
  const long                  dimensions[];  // nombre d'éléments dans le tenseur
 };
```

OnnxMapTypeInfo

La structure décrit la carte obtenue dans le paramètre de sortie d'un modèle ONNX

```
struct OnnxMapTypeInfo
 {
  const ENUM_ONNX_DATA_TYPE  key_type;      // type clé
  const OnnxTypeInfo&        value_type;    // type de valeur
 };
```

OnnxSequenceTypeInfo

La structure décrit la séquence obtenue dans le paramètre de sortie d'un modèle ONNX

© 2000-2024, MetaQuotes Ltd.

```
struct OnnxSequenceTypeInfo
 {
  const OnnxTypeInfo&        value_type;       // type de données dans la séquence
 };
```

ENUM_ONNX_TYPE

L'énumération **ENUM_ONNX_TYPE** décrit le type de paramètre d'un modèle

ID	Description
ONNX_TYPE_UNKNOWN	Inconnu
ONNX_TYPE_TENSOR	Tenseur
ONNX_TYPE_SEQUENCE	Séquence
ONNX_TYPE_MAP	Carte (map)
ONNX_TYPE_OPAQUE	Abstrait (opaque)
ONNX_TYPE_SPARSETENSOR	Tenseur creux (sparse tensor)

ENUM_ONNX_DATA_TYPE

L'énumération **ENUM_ONNX_DATA_TYPE** décrit le type de données utilisées

ID	Description
ONNX_DATA_TYPE_UNDEFINED	Indéfini
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_INT8	int 8 bits
ONNX_DATA_TYPE_UINT16	uint 16 bits
ONNX_DATA_TYPE_INT16	int 16 bits
ONNX_DATA_TYPE_INT32	int 32 bits
ONNX_DATA_TYPE_INT64	int 64 bits
ONNX_DATA_TYPE_STRING	string
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT16	float 16 bits
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	uint 32 bits
ONNX_DATA_TYPE_UINT64	uint 64 bits

© 2000-2024, MetaQuotes Ltd.

ID	Description
ONNX_DATA_TYPE_COMPLEX64	nombre complexe 64 bits
ONNX_DATA_TYPE_COMPLEX128	nombre complexe 128 bits
ONNX_DATA_TYPE_BFLOAT16	bfloat 16 bits (Brain Floating Point)

ENUM_ONNX_FLAGS

L'énumération **ENUM_ONNX_FLAGS** décrit le mode d'exécution du modèle

ID	Description
ONNX_DEBUG_LOGS	Journaux de débogage de sortie
ONNX_NO_CONVERSION	Désactive la conversion automatique, utiliser les données utilisateur telles quelles
ONNX_COMMON_FOLDER	Charge un fichier de modèle à partir du dossier Common\Files ; la valeur est égale au drapeau FILE_COMMON

© 2000-2024, MetaQuotes Ltd.

Bibliothèque Standard

Cet ensemble de chapitres contient les détails techniques de la Bibliothèque Standard MQL5 et la descriptions de ses composants principaux.

La Bibliothèque Standard MQL5 est écrite en MQL5 et est conçue pour faciliter l'écriture des programmes (indicateurs, scripts, experts) aux utilisateurs. La Bibliothèque fournit un accès aisé à la plupart des fonctions internes MQL5.

La Bibliothèque Standard MQL5 est située dans le répertoire de travail du terminal dans le répertoire Include.

Section	Localisation
Mathematics	Include\Math\
OpenCL	Include\OpenCL\
Classe de Base CObject	Include\
Collections de Données	Include\Arrays\
Collections de Données Génériques	Include\Generic\
Fichiers	Include\Files\
Chaînes de Caractères	Include\Strings\
Objets Graphiques	Include\Objects\
Objets Graphiques Personnalisés	Include\Canvas\
Graphiques 3D	Include\Canvas\
Graphiques des prix	Include\Charts\
Scientific Charts	Include\Graphics\
Indicateurs	Include\Indicators\
Classes pour le Trading	Include\Trade\
Modules de Stratégies	Include\Expert\
Panneaux et Dialogues	Include\Controls\

© 2000-2024, MetaQuotes Ltd.

Mathematiques

Plusieurs bibliothèques sont fournis pour effectuer des calculs dans divers domaine des mathématiques :

- Statistiques – fonctions pour travailler avec diverses distributions de la théorie des Probabilités
- Logique floue – bibliothèque qui implémente les systèmes d'interférences de Mamdani et de Sugeno
- ALGLIB – analyse de données (clustering, arbres de décision, régression linéaire, réseaux neuronaux), résolutions d'équations différentielles, transformation de Fourier, intégration numérique, problèmes d'optimisation, analyses statistiques, et plus.

© 2000-2024, MetaQuotes Ltd.

Statistiques

La Bibliothèque Statistique permet de travailler facilement avec les distributions statistiques de base.

La bibliothèque fournit 5 fonctions pour chaque distribution :

1. Calcul de la densité de probabilité - fonctions de la forme MathProbabilityDensityX()
2. Calcul des probabilités - fonctions de la forme MathCumulativeDistributionX()
3. Calcul des quantiles de la distribution - fonctions de la forme MathQuantileX()
4. Génération de nombres aléatoires avec la distribution spécifiée - fonctions de la forme MathRandomX()
5. Calcul des moments théoriques des distributions - fonctions de la forme MathMomentsX()

En complément du calcul des valeurs de chaque variable aléatoire, la bibliothèque implémente également des surcharges des fonctions rpour effedtuer les mêmes calculs pour des tableaux.

- Caractéristiques Statistiques
- Distribution Normale
- Distribution log-normale
- Distribution beta
- Distribution bêta non centrale
- Distribution gamma
- Distribution khi-deux
- Distribution khi-deux
- Distribution khi-deux non-centrale
- Distribution exponentielle
- Distribution de Fisher
- Distribution de Fisher non-centrale
- Distribution t
- Distribution logistique
- Distribution t non-centrale
- Distribution de Cauchy
- Distribution uniforme
- Distribution Weibull
- Distribution binomiale négative
- Distribution binomiale
- Distribution géométrique
- Distribution Hypergéometrique
- Distributio de Poisson
- Sous-fonctions

Exemple :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                   NormalDistributionExample.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- include the functions for calculating the normal distribution
#include <Math\Stat\Normal.mqh>
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- définit les paramètres de la distribution normale
   double mu=5.0;
   double sigma=1.0;
   PrintFormat("Distribution normale avec les paramètres mu=%G and sigma=%G, exemples
//--- définit l'intervalle
   double x1=mu-sigma;
   double x2=mu+sigma;
//--- variables pour le calcul de la probabilité
   double cdf1,cdf2,probability;
//--- variables pour les codes d'erreurs
   int error_code1,error_code2;
//--- calcule les valeurs des fonctions de distribution
   cdf1=MathCumulativeDistributionNormal(x1,mu,sigma,error_code1);
   cdf2=MathCumulativeDistributionNormal(x2,mu,sigma,error_code2);
//--- vérifie les codes d'erreur
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- calcule la probabilité d'une variable aléatoire dans l'intervalle
      probability=cdf2-cdf1;
      //--- sort le résultat
      PrintFormat("1. Calcule la probabilité d'une variable aléatoire dans l'intervall
      PrintFormat("  Réponse : Probabilité = %5.8f",probability);
     }

//--- Cherche l'intervalle de valeurs de la variable aléatoire x, correspondant aux 95
   probability=0.95; //  définit la probabilité de confiance
//--- définit les probabilités aux limites de l'intervalle
   double p1=(1.0-probability)*0.5;
   double p2=probability+(1.0-probability)*0.5;
//--- calcule les limites de l'intervalle
   x1=MathQuantileNormal(p1,mu,sigma,error_code1);
   x2=MathQuantileNormal(p2,mu,sigma,error_code2);
//--- vérifie les codes d'erreur
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- affiche le résultat
      PrintFormat("2. Pour l'intervalle de confiance = %.2f, cherche l'intervalle de l
      PrintFormat("  Réponse: l'intervalle est %5.8f <= x <=%5.8f",x1,x2);
     }

   PrintFormat("3. Calcule les 4 premiers moments théoriques de la distribution");
//--- Génère un tableau de nombres aléatoires, calcule les 4 premiers moments et les c
   int data_count=1000000;  // définit le nombre de valeurs et prépare un tableau
   double data[];
   ArrayResize(data,data_count);
//--- génère des valeurs aléatoires et les stocke dans le tableau
```

```
   for(int i=0; i<data_count; i++)
     {
      data[i]=MathRandomNormal(mu,sigma,error_code1);
     }
//--- définit l'indice de la valeur initiale et la quantité de données pour le calcul
   int start=0;
   int count=data_count;
//--- calcule les 4 premiers moments des valeurs générées
   double mean=MathMean(data,start,count);
   double variance=MathVariance(data,start,count);
   double skewness=MathSkewness(data,start,count);
   double kurtosis=MathKurtosis(data,start,count);
//--- variables pour les moments théoriques
   double normal_mean=0;
   double normal_variance=0;
   double normal_skewness=0;
   double normal_kurtosis=0;
//--- affiche les valeurs des moments calculés
   PrintFormat("            Moyenne         Variance          Coef Dissymetrie     Kur
   PrintFormat("Calculés  %.10f   %.10f      %.10f      %.10f",mean,variance,skewness,
//--- calcule les valeurs théoriques des moments et les compare avec les valeurs obter
   if(MathMomentsNormal(mu,sigma,normal_mean,normal_variance,normal_skewness,normal_ku
     {
      PrintFormat("Théoriques %.10f   %.10f      %.10f       %.10f",normal_mean,normal
      PrintFormat("Différence  %.10f   %.10f      %.10f       %.10f",mean-normal_mean,
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Caractéristiques Statistiques

Ce groupe de fonction calcule les Caractéristiques Statistiques des éléments d'un tableau :

- moyenne,
- variance,
- coefficient de dissymetrie,
- kurtosis,
- médiane,
- racine-moyenne-carrée et
- écart type.

Fonction	Description
MathMean	Calcule la moyenne (premier moment) des éléments d'un tableau
MathVariance	Calcule la variance (deuxième moment) des éléments d'un tableau
MathSkewness	Calcule le coefficient de dissymétrie (troisième moment) des éléments d'un tableau
MathKurtosis	Calcule le kurtosis (quatrième moment) des élémens d'un tableau
MathMoments	Calcule les 4 premiers moments (moyenne, variance, coefficient de dissymétrie, kurtosis) des éléments d'un tableau
MathMedian	Calcule la valeur moyenne des éléments d'un tableau
MathStandardDeviation	Calcule l'écart type des éléments d'un tableau
MathAverageDeviation	Calcule l'écart type moyen absolu des éléments du tableau

© 2000-2024, MetaQuotes Ltd.

MathMean

Calcule la moyenne (premier moment) des éléments du tableau. Analogue à mean() dans R.

```
double  MathMean(
   const double&  array[]                // tableau de données
   );
```

Paramètres

array

[in] Tableau des données pour le calcul de la moyenne.

start=0

[in] Indice de départ pour le calcul.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour le calcul.

Valeur de Retour

La moyenne des éléments du tableau. En cas d'erreur, retourne NaN (not a number).

© 2000-2024, MetaQuotes Ltd.

MathVariance

Calcule la variance (deuxième moment) des éléments du tableau. Analogue à var() dans R.

```
double  MathVariance(
   const double&  array[]                // tableau de données
   );
```

Paramètres

array

[in] Tableau de données pour les calculs.

start=0

[in] Indice de départ pour le calcul.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour le calcul.

Valeur de Retour

La variance des éléments du tableau. En cas d'erreur, retourne NaN (not a number).

© 2000-2024, MetaQuotes Ltd.

MathSkewness

Calcule le coefficient de dissymétrie (skewness en anglais) (troisième moment) des éléments du tableau. Analogue à skewness() dans R (bilbiothèque e1071).

```
double  MathSkewness(
   const double&  array[]                // tableau de données
   );
```

Paramètres

array

[in] Tableau de données pour les calculs.

start=0

[in] Indice de départ pour le calcul.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour le calcul.

Valeur de Retour

Le coefficient de dissymétrie des éléments du tableau. En cas d'erreur, retourne NaN (not a number).

© 2000-2024, MetaQuotes Ltd.

MathKurtosis

Calcule le kurtosis (quatrième moment) des éléments du tableau. Analogue à kurtosis() dans R (bilbiothèque e1071).

```
double  MathKurtosis(
   const double&  array[]                // tableau de données
   );
```

Paramètres

array

[in] Tableau de données pour les calculs.

start=0

[in] Indice de départ pour le calcul.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour le calcul.

Valeur de Retour

Le kurtosis des éléments du tableau. En cas d'erreur, retourne NaN (not a number).

Avertissement

Le calcul du kurtosis est effectué en utilisant le kurtosis en excès autour de la distribution normale (kurtosis en excès=kurtosis-3), c'est à dire que le kurtosis en excès de la distribution normale est zéro.

Il est positif si le pic de la distribution autour de la valeur attendue est pointu, et négatif si le pic est plat.

© 2000-2024, MetaQuotes Ltd.

MathMoments

Calcule les 4 premiers moments (moyenne, variance, coefficient de dissymétrie, kurtosis) des éléments du tableau.

```
double  MathMoments(
   const double&  array[],              // tableau de données
   double&        mean,                 // moyenn (1er moment)
   double&        variance,             // variance (2ème moment)
   double&        skewness,             // coef. de dissymétrie (3ème moment)
   double&        kurtosis,             // kurtosis (4ème moment)
   const int      start=0,              // indice de départ
   const int      count=WHOLE_ARRAY     // nombre d'éléments
   );
```

Paramètres

array

[in] Tableau de données pour les calculs.

mean

[out] Variable pour la moyenne (1er moment)

variance

[out] Variable pour la variance (2ème moment)

skewness

[out] Variable pour le coefficient de dissymétrie (3ème moment)

kurtosis

[out] Variable pour le kurtosis (4ème moment)

start=0

[in] Indice de départ pour le calcul.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour le calcul.

Valeur de Retour

Retourne true si les moments ont été calculés avec succès, sinon false.

Avertissement

Le calcul du kurtosis est effectué en utilisant le kurtosis en excès autour de la distribution normale (kurtosis en excès=kurtosis-3), c'est à dire que le kurtosis en excès de la distribution normale est zéro.

Il est positif si le pic de la distribution autour de la valeur attendue est pointu, et négatif si le pic est plat.

© 2000-2024, MetaQuotes Ltd.

MathMedian

Calcule la valeur médiane des éléments du tableau. Analogue à median() dans R.

```
double  MathMedian(
   const double&  array[]                // tableau de données
   );
```

Paramètres

array

[in] Tableau de données pour les calculs.

start=0

[in] Indice de départ pour le calcul.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour le calcul.

Valeur de Retour

La valeur médiane des éléments du tableau. En cas d'erreur, retourne NaN (not a number).

© 2000-2024, MetaQuotes Ltd.

MathStandardDeviation

Calcule l'écart-type des éléments du tableau. Analogue à sd() dans R.

```
double  MathStandardDeviation(
   const double&  array[]                // tableau de données
   );
```

Paramètres

array

[in] Tableau de données pour les calculs.

start=0

[in] Indice de départ pour le calcul.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour le calcul.

Valeur de Retour

L'écart type des éléments du tableau. En cas d'erreur, retourne NaN (not a number).

© 2000-2024, MetaQuotes Ltd.

MathAverageDeviation

Calcule l'écart absolu moyen des éléments du tableau. Analogue à aad() dans R.

```
double  MathAverageDeviation(
   const double&  array[]                // tableau de données
   );
```

Paramètres

array

[in] Tableau de données pour les calculs.

start=0

[in] Indice de départ pour le calcul.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour le calcul.

Valeur de Retour

L'écart absolu moyen des éléments du tableau. En cas d'erreur, retourne NaN (not a number).

© 2000-2024, MetaQuotes Ltd.

Distribution Normale

Cette section contient les fonctions utilisées pour travailler avec une distribution normale. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi normale. La distribution est définie par la formule suivante :

$$f_{Normal}(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Avec :

- x — valeur de la variable aléatoire
- μ — valeur attendue
- σ — écart quadratique moyen

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityNormal	Calcule la fonction de densité de probabilité de la distribution normale
MathCumulativeDistributionNormal	Calcule la valeur de la fonction de distribution de probabilité normale
MathQuantileNormal	Calcule la valeur de la fonction inverse de distribution normale pour la probabilité spécifiée
MathRandomNormal	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi normale
MathMomentsNormal	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution normale

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNormal

Calcule la valeur de la fonction de densité de probabilité de la distribution normale avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNormal(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // paramètre moyen de la distribution (valeur attendu
   const double  sigma,          // paramètre sigma de la distribution (écart quadrati
   const bool    log_mode,       // calcule le logarithme de la valeur
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution normale avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNormal(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // paramètre moyen de la distribution (valeur attendu
   const double  sigma,          // paramètre sigma de la distribution (écart quadrati
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité des probabilités de la distribution normale avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dnorm() dans R.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],            // tableau des valeurs de la variable aléatoire
   const double   mu,             // paramètre moyen de la distribution (valeur attend
   const double  sigma,          // paramètre sigma de la distribution (écart quadrati
   const bool    log_mode,       // calcule le logarithme de la valeur
   double&        result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de densité des probabilités de la distribution normale avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],            // tableau des valeurs de la variable aléatoire
   const double   mu,             // paramètre moyen de la distribution (valeur attend
   const double  sigma,          // paramètre sigma de la distribution (écart quadrati
   double&        result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

mu

[in] paramètre moyen de la distribution (valeur attendue).

sigma

[in] paramètre sigma de la distribution (écart quadratique moyen).

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau pour obtenir les valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNormal

Calcule la valeur de la fonction de distribution normale avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   const bool    tail,           // flag de calcul de la queue
   const bool    log_mode,       // calcule le logarithme de la valeur
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution normale avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution normale avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pnorm() dans R.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   const bool    tail,           // flag de calcul de la queue
   const bool    log_mode,       // calcule le logarithme de la valeur
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la valeur de la fonction de distribution normale avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

mu

[in] paramètre moyen de la distribution (valeur attendue).

sigma

[in] paramètre sigma de la distribution (écart quadratique moyen).

tail

[in] Flag de calcul. Si tail=true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau pour obtenir les valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNormal

Pour la *probabilitéy* spécifiée, la fonction calcule la valeur de la fonction de distribution normale inverse avec les paramètres mu et sigma. En cas d'erreur, retourne NaN.

```
double  MathQuantileNormal(
   const double  probability,    // probability valeur de la variable aléatoire
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   const bool    tail,           // flag de calcul de la queue
   const bool    log_mode,       // calcule le logarithme de la valeur
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilitéy* spécifiée, la fonction calcule la valeur de la fonction de distribution normale inverse avec les paramètres mu et sigma. En cas d'erreur, retourne NaN.

```
double  MathQuantileNormal(
   const double  probability,    // probability valeur de la variable aléatoire
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution normale inverse avec les paramètres mu et sigma. En cas d'erreur, retourne false. Analogue à qnorm() dans R.

```
bool  MathQuantileNormal(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   const bool    tail,           // flag de calcul de la queue
   const bool    log_mode,       // calcule le logarithme de la valeur
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution normale inverse avec les paramètres mu et sigma. En cas d'erreur, retourne false.

```
bool  MathQuantileNormal(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

© 2000-2024, MetaQuotes Ltd.

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

mu

[in] paramètre moyen de la distribution (valeur attendue).

sigma

[in] paramètre sigma de la distribution (écart quadratique moyen).

tail

[in] Flag de calcul. Si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau pour obtenir les quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomNormal

Génère une variable pseudo-aléatoire distribuée suivant la loi normale avec les paramètres mu et sigma. En cas d'erreur, retourne NaN.

```
double  MathRandomNormal(
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi normale avec les paramètres mu et sigma. En cas d'erreur, retourne false. Analogue à rnorm() dans R.

```
bool  MathRandomNormal(
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des variables pseudo-aléatoires
   );
```

Paramètres

mu

[in] paramètre moyen de la distribution (valeur attendue).

sigma

[in] paramètre sigma de la distribution (écart quadratique moyen).

data_count

[in] Le nombre de variables pseudo-aléatoires à obtenir.

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNormal

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution beta

```
double  MathMomentsNormal(
   const double  mu,             // valeur attendue
   const double  sigma,          // écart quadratique moyen
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable stockant le code d'erreur
   );
```

Paramètres

mu

[in] paramètre moyen de la distribution (valeur attendue).

sigma

[in] paramètre sigma de la distribution (écart quadratique moyen).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si les moments ont été calculés avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution log-normale

Cette section contient les fonctions utilisées pour travailler avec une distribution log-normale. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi log-normale. La distribution log-normale est définie par la formule suivante :

$$f_{Lognormal}(x \mid \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$$

Avec :

- x — valeur de la variable aléatoire
- μ — logarithme de la valeur attendue
- σ — logarithme de l'écart quadratique moyen

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityLognormal	Calcule la fonction de densité de probabilité de la distribution log-normale
MathCumulativeDistributionLognormal	Calcule la valeur de la fonction de distribution de probabilité log-normale
MathQuantileLognormal	Calcule la valeur de la fonction inverse de distribution log-normale pour la probabilité spécifiée
MathRandomLognormal	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi log-normale
MathMomentsLognormal	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution log-normale

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityLognormal

Calcule la valeur de la fonction de densité de probabilité de la distribution log-normale avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // logarithme de la valeur attendue (log moyen)
   const double  sigma,          // logarithme de l'écart quadratique moyen (log écart
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution log-normale avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // logarithme de la valeur attendue (log moyen)
   const double  sigma,          // logarithme de l'écart quadratique moyen (log écart
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité des probabilités de la distribution log-normale avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne NaN. Analogue à dlnorm() dans R.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],            // tableau des valeurs de la variable aléatoire
   const double   mu,             // logarithme de la valeur attendue (log moyen)
   const double   sigma,          // Logarithme de l'écart quadratique moyen (log écar
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   double&        result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de densité des probabilités de la distribution log-normale avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],            // tableau des valeurs de la variable aléatoire
   const double   mu,             // logarithme de la valeur attendue (log moyen)
   const double   sigma,          // Logarithme de l'écart quadratique moyen (log écar
   double&        result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

mu

[in] Logarithme de la valeur attendue (log_mean).

sigma

[in] Logarithme de l'écart quadratique moyen (log écart-type).

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau pour obtenir les valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionLognormal

Calcule la fonction de distribution log-normale des probabilités avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // logarithme de la valeur attendue (log moyen)
   const double  sigma,          // logarithme de l'écart quadratique moyen (log écart
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution log-normale des probabilités avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // logarithme de la valeur attendue (log moyen)
   const double  sigma,          // logarithme de l'écart quadratique moyen (log écart
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution log-normale des probabilités avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à plnorm() dans R.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],            // tableau des valeurs de la variable aléatoire
   const double   mu,             // logarithme de la valeur attendue (log moyen)
   const double   sigma,          // Logarithme de l'écart quadratique moyen (log écar
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&        result[]        // tableau des valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution log-normale des probabilités avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],            // tableau des valeurs de la variable aléatoire
   const double   mu,             // logarithme de la valeur attendue (log moyen)
   const double   sigma,          // Logarithme de l'écart quadratique moyen (log écar
   double&        result[]        // tableau des valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

mu

[in] Logarithme de la valeur attendue (log_mean).

sigma

[in] Logarithme de l'écart quadratique moyen (log écart-type).

tail

[in] Flag de calcul, si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau pour obtenir les valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileLognormal

Pour la probabilité spécifiée, la fonction calcule la valeur de la fonction de distribution log-normale inverse avec les paramètres mu et sigma. En cas d'erreur, retourne NaN.

```
double  MathQuantileLognormal(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  mu,             // logarithme de la valeur attendue (log moyen)
   const double  sigma,          // logarithme de l'écart quadratique moyen (log écart
   const bool    tail,           // flag de calcul, si false, alors le calcul est effec
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ei
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la probabilité spécifiée, la fonction calcule la valeur de la fonction de distribution log-normale inverse avec les paramètres mu et sigma. En cas d'erreur, retourne NaN.

```
double  MathQuantileLognormal(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  mu,             // logarithme de la valeur attendue (log moyen)
   const double  sigma,          // logarithme de l'écart quadratique moyen (log écart
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau probability[] spécifié, la fonction calcule la valeur de la fonction log-normale inverse avec les paramètres mu et sigma. En cas d'erreur, retourne false. Analogue à qlnorm() dans R.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // tableau des valeurs de probabilités de la variabl
   const double   mu,             // logarithme de la valeur attendue (log moyen)
   const double   sigma,          // Logarithme de l'écart quadratique moyen (log écar
   const bool     tail,           // flag de calcul, si false, alors le calcul est effe
   const bool     log_mode,       // flag de calcul, si log_mode=true, le calcul est ei
   double&        result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau probability[] spécifié, la fonction calcule la valeur de la fonction de distribution log-normale inverse avec les paramètres mu et sigma. En cas d'erreur, retourne false.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // tableau des valeurs de probabilités de la variabl
   const double   mu,             // logarithme de la valeur attendue (log moyen)
   const double   sigma,          // Logarithme de l'écart quadratique moyen (log écar
   double&        result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

© 2000-2024, MetaQuotes Ltd.

[in] Probability valeur de la variable aléatoire occurrence.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

mu

[in] Logarithme de la valeur attendue (log_mean).

sigma

[in] Logarithme de l'écart quadratique moyen (log écart-type).

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomLognormal

Génère une variable pseudo-aléatoire distribuée suivant la loi log-normale avec les paramètres mu et sigma. En cas d'erreur, retourne NaN.

```
double  MathRandomLognormal(
   const double  mu,             // logarithme de la valeur attendue (log moyen)
   const double  sigma,          // logarithme de l'écart quadratique moyen (log écart
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi log-normale avec les paramètres mu et sigma. En cas d'erreur, retourne false. Analogue à rlnorm() dans R.

```
double  MathRandomLognormal(
   const double  mu,             // logarithme de la valeur attendue (log moyen)
   const double  sigma,          // logarithme de l'écart quadratique moyen (log écart
   const int     data_count,     // quantité de données requises
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

mu

[in] Logarithme de la valeur attendue (log_mean).

sigma

[in] Logarithme de l'écart quadratique moyen (log écart-type).

data_count

[in] Quantité de données requises.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsLognormal

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution log-normale Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

```
double  MathMomentsLognormal(
   const double  mu,             // logarithme de la valeur attendue (log moyen)
   const double  sigma,          // logarithme de l'écart quadratique moyen (log écart
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour stocker le code d'erreur
   );
```

Paramètres

mu

[in] Logarithme de la valeur attendue (log_mean).

sigma

[in] Logarithme de l'écart quadratique moyen (log écart-type).

mean

[in] Variable pour la moyenne.

variance

[out] Variable pour la variance.

skewness

[out] Variable pour le coefficient de dissymétrie.

kurtosis

[out] Variable pour le kurtosis.

error_code

[out] Variable pour stocker le code d'erreur.

Valeur de Retour

Retourne true si les moments ont été calculés avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution beta

Cette section contient les fonctions permettant de travailler avec une distribution beta. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi correspondante. La distribution beta est définie avec la formule suivante :

$$f_{Beta}(x \mid a, b) = \frac{1}{B(a, b)} x^{a-1}(1-x)^{b-1}$$

Avec :

- x — valeur de la variable aléatoire
- a — le premier paramètre de la distribution beta
- b — le deuxième paramètre de la distribution beta

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityBeta	Calcule la fonction de densité de probabilité de la distribution beta
MathCumulativeDistributionBeta	Calcule la valeur de la fonction de distribution de probabilité beta
MathQuantileBeta	Calcule la valeur de la fonction inverse de distribution beta pour la probabilité spécifiée
MathRandomBeta	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de distribution beta
MathMomentsBeta	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution beta

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityBeta

Calcule la valeur de la fonction de densité de probabilité de la distribution beta avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityBeta(
   const double  x,              // valeur de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shape
   const double  b,              // le deuxième paramètre de la distribution beta (shap
   const bool    log_mode,        // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution beta avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityBeta(
   const double  x,              // valeur de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shape
   const double  b,              // le deuxième paramètre de la distribution beta (shap
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution beta avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dbeta() dans R.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution beta avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

a

[in] Le premier paramètre de la distribution beta (shape 1).

b

[in] Le deuxième paramètre de la distribution beta (shape 2)

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionBeta

Calcule la fonction de distribution de probabilité de la distribution beta avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // valeur de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shape
   const double  b,              // le deuxième paramètre de la distribution beta (shap
   const bool    tail,           // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,        // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution beta avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // valeur de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shape
   const double  b,              // le deuxième paramètre de la distribution beta (shap
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution beta avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pbeta() dans R.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],             // tableau de valeurs de la variable aléatoire
   const double  a,               // le premier paramètre de la distribution beta (shap
   const double  b,               // le deuxième paramètre de la distribution beta (sha
   const bool    tail,           // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,        // flag pour calculer le logarithme de la valeur, si
   double&       result[]         // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution de probabilité de la distribution beta avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],             // tableau de valeurs de la variable aléatoire
   const double  a,               // le premier paramètre de la distribution beta (shap
   const double  b,               // le deuxième paramètre de la distribution beta (sha
   double&       result[]         // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

a

[in] Le premier paramètre de la distribution beta (shape 1).

b

[in] Le deuxième paramètre de la distribution beta (shape 2)

tail

[in] Flag de calcul, si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileBeta

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction inverse de distribution beta avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathQuantileBeta(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction inverse de distribution beta avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathQuantileBeta(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau probability[] spécifié, la fonction calcule les valeurs de la fonction de distribution beta inverse avec les paramètres a et b En cas d'erreur, retourne false. Analogue à qbeta() dans R.

```
double  MathQuantileBeta(
   const double&  probability[], // tableau des valeurs de probabilités de la variable
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau probability[] spécifié, la fonction calcule les valeurs de la fonction de distribution beta inverse avec les paramètres a et b En cas d'erreur, retourne false.

```
bool  MathQuantileBeta(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

a

[in] Le premier paramètre de la distribution beta (shape1).

b

[in] Le deuxième paramètre de la distribution beta (shape2).

tail

[in] Flag de calcul, si lower_tail=false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomBeta

Génère une variable pseudo-aléatoire distribuée suivant la loi de distribution beta avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathRandomBeta(
   const double  a,             // le premier paramètre de la distribution beta (shape
   const double  b,             // le deuxième paramètre de la distribution beta (shap
   int&          error_code     // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi de distribution beta avec les paramètres a et b. En cas d'erreur, retourne false. Analogue à rbeta() dans R.

```
bool  MathRandomBeta(
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des variables pseudo-aléatoires
   );
```

Paramètres

a

[in] Le premier paramètre de la distribution beta (shape1)

b

[in] Le deuxième paramètre de la distribution beta (shape2).

data_count

[in] Le nombre de variables pseudo-aléatoires à obtenir.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsBeta

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution beta

```
double  MathMomentsBeta(
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewn
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

a

[in] Le premier paramètre de la distribution beta (shape1).

b

[in] Le deuxième paramètre de la distribution beta (shape2).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si les moments ont été calculés avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution bêta non centrale

Cette section contient les fonctions permettant de travailler avec une distribution bêta non centrale. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi correspondante. La distribution bêta non centrale est définie avec la formule suivante :

$$f_{NoncentralBeta}(x \mid a, b, \lambda) = \sum_{r=0}^{\infty} e^{-\frac{\lambda}{2}} \frac{\left(\frac{\lambda}{2}\right)^r}{r!} \frac{x^{a+r-1}(1-x)^{b-1}}{B(a+r, b)}$$

Avec :

- x — valeur de la variable aléatoire
- a — le premier paramètre de la distribution beta
- b — le deuxième paramètre de la distribution beta
- λ — paramètre de noncentralité

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityNoncentralBeta	Calcule la fonction de densité de probabilité de la distribution bêta non centrale
MathCumulativeDistributionNoncentralBeta	Calcule la valeur de la fonction de distribution de probabilité bêta non centrale
MathQuantileNoncentralBeta	Calcule la valeur de la fonction inverse de distribution bêta non centrale pour la probabilité spécifiée
MathRandomNoncentralBeta	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de distribution bêta non centrale
MathMomentsNoncentralBeta	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution bêta non centrale

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNoncentralBeta

Calcule la valeur de la fonction de densité de probabilité de la distribution bêta non centrale avec les paramètres a, b et lambda pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // le premier paramètre de la distribution beta (shape
   const double  b,             // le deuxième paramètre de la distribution beta (shap
   const double  lambda,        // paramètre de noncentralité
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution bêta non centrale avec les paramètres a, b et lambda pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // le premier paramètre de la distribution beta (shape
   const double  b,             // le deuxième paramètre de la distribution beta (shap
   const double  lambda,        // paramètre de noncentralité
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution bêta non centrale avec les paramètres a, b et lambda pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dbeta() dans R.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const double  lambda,         // paramètre de noncentralité
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution beta avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const double  lambda,         // paramètre de noncentralité
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Paramètres

© 2000-2024, MetaQuotes Ltd.

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

a

[in] Le premier paramètre de la distribution beta (shape 1).

b

[in] Le deuxième paramètre de la distribution beta (shape 2)

lambda

[in] Paramètre de noncentralité

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralBeta

Calcule la fonction de distribution de probabilités de la distribution bêta non centrale avec les paramètres a, b et lambda pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // le premier paramètre de la distribution beta (shape
   const double  b,             // le deuxième paramètre de la distribution beta (shap
   const double  lambda,        // paramètre de noncentralité
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilités de la distribution bêta non centrale avec les paramètres a, b et lambda pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // le premier paramètre de la distribution beta (shape
   const double  b,             // le deuxième paramètre de la distribution beta (shap
   const double  lambda,        // paramètre de noncentralité
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilités de la distribution bêta non centrale avec les paramètres a, b et lambda pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pbeta() dans R.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const double  lambda,         // paramètre de noncentralité
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution de probabilités de la distribution bêta non centrale avec les paramètres a, b et lambda pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const double  lambda,         // paramètre de noncentralité
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

© 2000-2024, MetaQuotes Ltd.

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

a

[in] Le premier paramètre de la distribution beta (shape 1).

b

[in] Le deuxième paramètre de la distribution beta (shape 2)

lambda

[in] Paramètre de noncentralité

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNoncentralBeta

Calcule la valeur de la fonction de distribution des probabilités inverses de la distribution bêta non centrale avec les paramètres a, b et lambda pour la probabilité d'occurence d'une *probabilité* variable aléatoire. En cas d'erreur, retourne NaN.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const double  lambda,         // paramètre de noncentralité
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution des probabilités inverses de la distribution bêta non centrale avec les paramètres a, b et lambda pour la probabilité d'occurence d'une *probabilité* variable aléatoire. En cas d'erreur, retourne NaN.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const double  lambda,         // paramètre de noncentralité
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau probability[] spécifié, la fonction calcule la valeur de la fonction de distribution inverse de la distribution beta non centrale avec les paramètres a, b et lambda En cas d'erreur, retourne false. Analogue à qbeta() dans R.

```
double  MathQuantileNoncentralBeta(
   const double&  probability[],  // tableau des valeurs de probabilités de la variabl
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const double  lambda,         // paramètre de noncentralité
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau probability[] spécifié, la fonction calcule la valeur de la fonction de distribution de probabilité inverse de la distribution beta non centrale avec les paramètres a, b et lambda. En cas d'erreur, retourne false.

```
bool  MathQuantileNoncentralBeta(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
```

© 2000-2024, MetaQuotes Ltd.

```
   const double  lambda,          // paramètre de noncentralité
   double&       result[]         // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

a

[in] Le premier paramètre de la distribution beta (shape1).

b

[in] Le deuxième paramètre de la distribution beta (shape2).

lambda

[in] Paramètre de noncentralité.

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomNoncentralBeta

Génère une variable pseudo-aléatoire distribuée suivant la loi de distribution bêta non centrale avec les paramètres a, b et lambda. En cas d'erreur, retourne NaN.

```
double  MathRandomNoncentralBeta(
   const double  a,             // le premier paramètre de la distribution beta (shape
   const double  b,             // le deuxième paramètre de la distribution beta (shap
   const double  lambda,        // paramètre de noncentralité
   int&          error_code     // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi de distribution bêta non centrale avec les paramètres a, b et lambda. En cas d'erreur, retourne false. Analogue à rbeta() dans R.

```
bool  MathRandomNoncentralBeta(
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const double  lambda,         // paramètre de noncentralité
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des variables pseudo-aléatoires
   );
```

Paramètres

a

[in] Le premier paramètre de la distribution beta (shape1)

b

[in] Le deuxième paramètre de la distribution beta (shape2).

lambda

[in] Paramètre de noncentralité

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNoncentralBeta

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution bêta non centrale avec les paramètres a et b.

```
double  MathMomentsNoncentralBeta(
   const double  a,              // le premier paramètre de la distribution beta (shap
   const double  b,              // le deuxième paramètre de la distribution beta (sha
   const double  lambda,         // paramètre de noncentralité
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewn
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

a

[in] Le premier paramètre de la distribution beta (shape1).

b

[in] Le deuxième paramètre de la distribution beta (shape2).

lambda

[in] Paramètre de noncentralité

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution gamma

Cette section contient les fonctions permettant de travailler avec une distribution gamma. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi correspondante. La distribution gamma est définie par la formule suivante :

$$f_{Gamma}(x \mid a, b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-\frac{x}{b}}$$

Avec :

- x — valeur de la variable aléatoire
- a — le premie paramètre de la distribution
- b — le deuxième paramètre de la distribution

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityGamma	Calcule la fonction de densité de probabilité de la distribution gamma
MathCumulativeDistributionGamma	Calcule la valeur de la fonction de distribution de probabilité gamma
MathQuantileGamma	Calcule la valeur de la fonction de distribution gamme inverse pour la probabilité spécifiée
MathRandomGamma	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de distribution gamma
MathMomentsGamma	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution gamma

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityGamma

Calcule la valeur de la fonction de densité de probabilité de la distribution gamma avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityGamma(
   const double  x,              // valeur de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=t
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution gamma avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityGamma(
   const double  x,              // valeur de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution gamma avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dgamma() dans R.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution gamma avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

a

[in] Le premier paramètre de la distribution (forme).

b

[in] Le second paramètre de la distribution (échelle).

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionGamma

Calcule la fonction de distribution de probabilité de la distribution gamma avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // valeur de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   const bool    tail,           // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution gamma avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // valeur de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution gamma avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pgamma() dans R.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution de probabilité de la distribution gamma avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

x[]

[in] Tableau des valeurs de la variable aléatoire.

a

[in] Le premier paramètre de la distribution (forme).

b

[in] le second paramètre de la distribution (échelle)

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileGamma

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de distribution gamma inverse avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathQuantileGamma(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de distribution gamma inverse avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathQuantileGamma(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution gamma inverse avec les paramètres a et b En cas d'erreur, retourne false. Analogue à qgamma() dans R.

```
double  MathQuantileGamma(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution gamma inverse avec les paramètres a et b En cas d'erreur, retourne false.

```
bool  MathQuantileGamma(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

a

[in] Le premier paramètre de la distribution (forme).

b

[in] Le second paramètre de la distribution (échelle).

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomGamma

Génère une variable pseudo-aléatoire distribuée suivant la loi de la distribution gamma avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathRandomGamma(
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi de la distribution gamma avec les paramètres a et b. En cas d'erreur, retourne false. Analogue à rgamma() dans R.

```
bool  MathRandomGamma(
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

a

[in] Le premier paramètre de la distribution (forme).

b

[in] Le second paramètre de la distribution (échelle).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsGamma

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution gamma avec les paramètres a et b.

```
double  MathMomentsGamma(
   const double  a,              // le premier paramètre de la distribution (forme)
   const double  b,              // le second paramètre de la distribution (échelle)
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

a

[in] // le premier paramètre de la distribution (forme).

b

[in] le second paramètre de la distribution (échelle).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution khi-deux

Cette section contient les fonctions permettant de travailler avec une distribution khi-deux. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi correspondante. La distribution khi-deux est définie par la formule suivante :

$$f_{Chi-Square}(x|\nu) = \frac{x^{\frac{(\nu-2)}{2}} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}} \Gamma\left(\frac{\nu}{2}\right)}$$

Avec :

- x — valeur de la variable aléatoire
- ν — nombre de degrés de liberté

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityChiSquare	Calcule la fonction de densité de probabilité de la distribution khi-deux
MathCumulativeDistributionChiSquare	Calcule la valeur de la fonction de distribution de probabilité khi-deux
MathQuantileChiSquare	Calcule la valeur de la fonction inverse de distribution de khi-deux pour la probabilité spécifiée
MathRandomChiSquare	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de distribution khi-deux
MathMomentsChiSquare	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution khi-deux

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityChiSquare

Calcule la fonction de densité de probabilité de la distribution khi-deux avec le paramètre nu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // valeur de la variable aléatoire
   const double  nu,             // paramètre de la distribution (nombre de degrés de l
   const bool    log_mode,        // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution khi-deux avec le paramètre nu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // valeur de la variable aléatoire
   const double  nu,             // paramètre de la distribution (nombre de degrés de l
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution khi-deux avec le paramètre nu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dchisq() dans R.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],             // tableau de valeurs de la variable aléatoire
   const double  nu,             // paramètre de la distribution (nombre de degrés de l
   const bool    log_mode,        // flag pour calculer le logarithme de la valeur, si
   double&       result[]         // tableau de valeurs de la fonction de densité de pr
   );
```

Calcule la fonction de densité de probabilité de la distribution khi-deux avec le paramètre nu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],             // tableau de valeurs de la variable aléatoire
   const double  nu,             // paramètre de la distribution (nombre de degrés de l
   double&       result[]         // tableau de valeurs de la fonction de densité de pr
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

log_mode

© 2000-2024, MetaQuotes Ltd.

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionChiSquare

Calcule la fonction de distribution de probabilité de la distribution khi-deux avec le paramètre nu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // valeur de la variable aléatoire
   const double  nu,             // paramètre de la distribution (nombre de degrés de l
   const bool    tail,           // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,        // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution khi-deux avec le paramètre nu pour une variable aléatoire x. En cas d'erreur, retourne NaN

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // valeur de la variable aléatoire
   const double  nu,             // paramètre de la distribution (nombre de degrés de l
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution khi-deux avec le paramètre nu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pchisq() dans R.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],             // tableau de valeurs de la variable aléatoire
   const double  nu,             // paramètre de la distribution (nombre de degrés de l
   const bool    tail,            // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,        // flag pour calculer le logarithme de la valeur, si
   double&       result[]         // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution de probabilité de la distribution khi-deux avec le paramètre nu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],             // tableau de valeurs de la variable aléatoire
   const double  nu,             // paramètre de la distribution (nombre de degrés de l
   double&       result[]         // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileChiSquare

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction inverse de distribution khi-deux. En cas d'erreur, retourne NaN.

```
double  MathQuantileChiSquare(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction inverse de distribution khi-deux. En cas d'erreur, retourne NaN.

```
double  MathQuantileChiSquare(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution khi-deux inverse. En cas d'erreur, retourne false. Analogue à qchisq() dans R.

```
double  MathQuantileChiSquare(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution khi-deux inverse. En cas d'erreur, retourne false.

```
bool  MathQuantileChiSquare(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomChiSquare

Génère une variable pseudo-alatoire distribuée suivant la loi de distribution khi-deux avec le paramètre nu. En cas d'erreur, retourne NaN.

```
double  MathRandomChiSquare(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-alatoires distribuées suivant la loi de distribution khi-deux avec le paramètre nu. En cas d'erreur, retourne false. Analogue à rchisq() dans R.

```
bool  MathRandomChiSquare(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsChiSquare

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution khi-deux avec le paramètre nu

```
double  MathMomentsChiSquare(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewn
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution khi-deux non-centrale

Cette section contient les fonctions permettant de travailler avec une distribution khi-deux non-centrale. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi correspondante. La distribution khi-deux non-centrale est définie par la formule suivante :

$$f_{NoncentralChiSquare}(x|\nu,\sigma)=\frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{1}{2}\right)}x^{\frac{\nu}{2}-1}e^{-\frac{(x+\sigma)}{2}}\sum_{r=0}^{\infty}\frac{(\lambda x)^r}{(2r)!}\frac{\Gamma\left(\frac{1}{2}+r\right)}{\Gamma\left(\frac{\nu}{2}+r\right)}$$

Avec :

- x — valeur de la variable aléatoire
- ν — nombre de degrés de liberté
- σ — paramètre de noncentralité

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityNoncentralChiSquare	Calcule la fonction de densité de probabilité de la distribution khi-deux non-centrale
MathCumulativeDistributionNoncentralChiSquare	Calcule la valeur de la fonction de distribution de probabilité khi-deux non-centrale
MathQuantileNoncentralChiSquare	Calcule la valeur de la fonction inverse de distribution de khi-deux non-centrale pour la probabilité spécifiée
MathRandomNoncentralChiSquare	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de distribution khi-deux non-centrale
MathMomentsNoncentralChiSquare	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution khi-deux non-centrale

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNoncentralChiSquare

Calcule la fonction de densité de probabilité de la distribution khi-deux non-centrale avec les paramètres nu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,         // paramètre de noncentralité
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution khi-deux non-centrale avec les paramètres nu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,         // paramètre de noncentralité
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution khi-deux non-centrale avec les paramètres nu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dchisq() dans R.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,          // paramètre de noncentralité
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Calcule la fonction de densité de probabilité de la distribution khi-deux avec le paramètre nu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,          // paramètre de noncentralité
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

sigma

[in] Paramètre de noncentralité.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralChiSquare

Calcule la fonction de distribution de probabilité de la distribution khi-deux non-centrale avec les paramètres nu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,         // paramètre de noncentralité
   const bool    tail,          // flag de calcul, si lower_tail=true, alors la probab
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution khi-deux non-centrale avec les paramètres nu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,         // paramètre de noncentralité
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution khi-deux non-centrale avec les paramètres nu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pchisq() dans R.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,          // paramètre de noncentralité
   const bool    tail,          // flag de calcul, si lower_tail=true, alors la probab
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution de probabilité de la distribution khi-deux non-centrale avec les paramètres nu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,          // paramètre de noncentralité
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

x[]

[in] Tableau des valeurs de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

sigma

[in] Paramètre de noncentralité.

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNoncentralChiSquare

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de distribution khi-deux non-centrale inverse avec les paramètres nu et sigma. En cas d'erreur, retourne NaN.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,          // paramètre de noncentralité
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de distribution khi-deux non-centrale inverse avec les paramètres nu et sigma. En cas d'erreur, retourne NaN.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,          // paramètre de noncentralité
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution khi-deux non centrale inverse avec les paramètres nu et sigma. En cas d'erreur, retourne false. Analogue à qchisq() dans R.

```
double  MathQuantileNoncentralChiSquare(
   const double&  probability[],  // tableau des valeurs de probabilités de la variabl
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,           // paramètre de noncentralité
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution khi-deux non centrale inverse. En cas d'erreur, retourne false.

```
bool  MathQuantileNoncentralChiSquare(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,          // paramètre de noncentralité
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

sigma

[in] Paramètre de noncentralité.

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomNoncentralChiSquare

Génère une variable pseudo-alatoire distribuée suivant la loi de distribution khi-deux non-centrale avec les paramètres nu et sigma. En cas d'erreur, retourne NaN.

```
double  MathRandomNoncentralChiSquare(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,         // paramètre de noncentralité
   int&          error_code     // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-alatoires distribuées suivant la loi de distribution khi-deux non-centrale avec les paramètres nu et sigma. En cas d'erreur, retourne false. Analogue à rchisq() dans R.

```
bool  MathRandomNoncentralChiSquare(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,          // paramètre de noncentralité
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

sigma

[in] Paramètre de noncentralité.

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNoncentralChiSquare

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution khi-deux non-centrale avec les paramètres nu et sigma.

```
double  MathMomentsNoncentralChiSquare(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  sigma,          // paramètre de noncentralité
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

sigma

[in] Paramètre de noncentralité.

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution exponentielle

Cette section contient les fonctions permettant de travailler avec une distribution exponentielle. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi de distribution exponentielle. La distribution exponentielle est définie par la formule suivante :

$$f_{Exponential}(x \mid \mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

Avec :

- x — valeur de la variable aléatoire
- μ — valeur attendue

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityExponential	Calcule la fonction de densité de probabilité de la distribution exponentielle
MathCumulativeDistributionExponential	Calcule la valeur de la fonction de distribution de probabilité exponentielle
MathQuantileExponential	Calcule la valeur de la fonction inverse de distribution exponentielle pour la probabilité spécifiée
MathRandomExponential	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de distribution exponentielle
MathMomentsExponential	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution exponentielle

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityExponential

Calcule la fonction de densité de probabilité de la distribution exponentielle avec le paramètre mu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityExponential(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // paramètre de la distribution (valeur attendue)
   const bool    log_mode,        // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution exponentielle avec le paramètre mu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityExponential(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // paramètre de la distribution (valeur attendue)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution exponentielle avec le paramètre mu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dexp() dans R.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  mu,             // paramètre de la distribution (valeur attendue)
   const bool    log_mode,        // flag pour calculer le logarithme de la valeur, si
   double&       result[]         // tableau de valeurs de la fonction de densité de pr
   );
```

Calcule la fonction de densité de probabilité de la distribution exponentielle avec le paramètre mu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  mu,             // paramètre de la distribution (valeur attendue)
   double&       result[]         // tableau de valeurs de la fonction de densité de pr
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

mu

[in] Paramètre de la distribution (valeur attendue)

log_mode

© 2000-2024, MetaQuotes Ltd.

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionExponential

Calcule la fonction de distribution exponentielle de probabilités avec le paramètre mu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // paramètre de la distribution (valeur attendue)
   const bool    tail,           // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,        // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution exponentielle de probabilités avec le paramètre mu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // valeur de la variable aléatoire
   const double  mu,             // paramètre de la distribution (valeur attendue)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution exponentielle de probabilités avec le paramètre mu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pexp() dans R.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],             // tableau de valeurs de la variable aléatoire
   const double  mu,              // paramètre de la distribution (valeur attendue)
   const bool    tail,            // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,        // flag pour calculer le logarithme de la valeur, si
   double&       result[]         // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution exponentielle de probabilités avec le paramètre mu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],             // tableau de valeurs de la variable aléatoire
   const double  mu,              // paramètre de la distribution (valeur attendue)
   double&       result[]         // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

mu

[in] Paramètre de la distribution (valeur attendue).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileExponential

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction exponentielle inverse avec le paramètre mu. En cas d'erreur, retourne NaN.

```
double  MathQuantileExponential(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  mu,             // paramètre de la distribution (valeur attendue)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ei
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction exponentielle inverse avec le paramètre mu. En cas d'erreur, retourne NaN.

```
double  MathQuantileExponential(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  mu,             // paramètre de la distribution (valeur attendue)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution exponentielle inverse avec le paramètre mu. En cas d'erreur, retourne false. Analogue à qexp() dans R.

```
double  MathQuantileExponential(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  mu,             // paramètre de la distribution (valeur attendue)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ei
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution exponentielle inverse avec le paramètre mu. En cas d'erreur, retourne false.

```
bool  MathQuantileExponential(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  mu,             // paramètre de la distribution (valeur attendue)
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

mu

© 2000-2024, MetaQuotes Ltd.

[in] Paramètre de la distribution (valeur attendue).

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomExponential

Génère une variable pseudo-alatoire distribuée suivant la loi de distribution exponentielle avec le paramètre mu. En cas d'erreur, retourne NaN.

```
double  MathRandomExponential(
   const double  mu,            // paramètre de la distribution (valeur attendue)
   int&          error_code     // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-alatoires distribuées suivant la loi de distribution exponentielle avec le paramètre mu. En cas d'erreur, retourne false. Analogue à rexp() dans R.

```
bool  MathRandomExponential(
   const double  mu,             // paramètre de la distribution (valeur attendue)
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

mu

[in] Paramètre de la distribution (valeur attendue).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsExponential

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution exponentielle avec le paramètre mu.

```
double  MathMomentsExponential(
   const double  mu,             // paramètre de la distribution (valeur attendue)
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

mu

[in] Paramètre de la distribution (valeur attendue).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution de Fisher

Cette section contient les fonctions permettant de travailler avec une distribution de Fisher. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi de Fisher. La distribution de Fisher est définie par la formule suivante :

$$f_F(x|\nu_1,\nu_2)=\frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}\right)}{\Gamma\left(\frac{\nu_1}{2}\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_1}{2}}\frac{x^{\frac{\nu_1-2}{2}}}{\left(1+\left(\frac{\nu_1}{\nu_2}\right)x\right)^{\frac{\nu_1+\nu_2}{2}}}$$

Avec :

- x — valeur de la variable aléatoire
- ν1 — le premier paramètre de distribution (nombre de degrés de liberté)
- ν2 — le deuxième paramètre de distribution (nombre de degrés de liberté)

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityF	Calcule la fonction de densité de probabilité de la distribution de Fisher
MathCumulativeDistributionF	Calcule la valeur de la fonction de la distribution de Fisher
MathQuantileF	Calcule la valeur de la fonction inverse de distribution de Fisher pour la probabilité spécifiée
MathRandomF	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de Fisher
MathMomentsF	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution de Fisher

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityF

Calcule la fonction de densité de probabilité de la distribution de Fisher avec les paramètres nu1 et nu2 pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityF(
   const double  x,              // valeur de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de deg
   const double  nu2,            // le deuxième paramètre de distribution (nombre de de
   const bool    log_mode,        // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher avec les paramètres nu1 et nu2 pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityF(
   const double  x,              // valeur de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de deg
   const double  nu2,            // le deuxième paramètre de distribution (nombre de de
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher avec les paramètres nu1 et nu2 pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à df() dans R.

```
bool  MathProbabilityDensityF(
   const double& x[],             // tableau de valeurs de la variable aléatoire
   const double  nu1,             // le premier paramètre de distribution (nombre de de
   const double  nu2,             // le deuxième paramètre de distribution (nombre de d
   const bool    log_mode,        // flag pour calculer le logarithme de la valeur, si
   double&       result[]         // tableau de valeurs de la fonction de densité de pr
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher avec les paramètres nu1 et nu2 pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityF(
   const double& x[],             // tableau de valeurs de la variable aléatoire
   const double  nu1,             // le premier paramètre de distribution (nombre de de
   const double  nu2,             // le deuxième paramètre de distribution (nombre de d
   double&       result[]         // tableau de valeurs de la fonction de densité de pr
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

nu1

[in] Le premier paramètre de distribution (nombre de degrés de liberté).

nu2

[in] Le deuxième paramètre de distribution (nombre de degrés de liberté).

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionF

Calcule la fonction de densité de probabilité de la distribution de Fisher avec les paramètres nu1 et nu2 pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionF(
   const double  x,             // valeur de la variable aléatoire
   const double  nu1,           // le premier paramètre de distribution (nombre de deg
   const double  nu2,           // le deuxième paramètre de distribution (nombre de de
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher avec les paramètres nu1 et nu2 pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionF(
   const double  x,             // valeur de la variable aléatoire
   const double  nu1,           // le premier paramètre de distribution (nombre de deg
   const double  nu2,           // le deuxième paramètre de distribution (nombre de de
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher avec les paramètres nu1 et nu2 pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pf() dans R.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher avec les paramètres nu1 et nu2 pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

x[]

[in] Tableau des valeurs de la variable aléatoire.

nu1

[in] Le premier paramètre de distribution (nombre de degrés de liberté).

nu2

[in] Le deuxième paramètre de distribution (nombre de degrés de liberté).

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileF

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de la distribution de Fisher inverse avec les paramètres nu1 et nu2. En cas d'erreur, retourne NaN.

```
double  MathQuantileF(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de la distribution de Fisher inverse avec les paramètres nu1 et nu2. En cas d'erreur, retourne NaN.

```
double  MathQuantileF(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution de Fisher inverse avec les paramètres nu1 et nu2 En cas d'erreur, retourne false. Analogue à qf() dans R.

```
double  MathQuantileF(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution de Fisher inverse avec les paramètres nu1 et nu2 En cas d'erreur, retourne false.

```
bool  MathQuantileF(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

© 2000-2024, MetaQuotes Ltd.

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

nu1

[in] Le premier paramètre de distribution (nombre de degrés de liberté).

nu2

[in] Le deuxième paramètre de distribution (nombre de degrés de liberté).

tail

[in] Flag de calcul, si lower_tail=false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomF

Génère une variable pseudo-alatoire distribuée suivant la loi de distribution de paramètremu avec les paramètres nu1 et nu2. En cas d'erreur, retourne NaN.

```
double  MathRandomF(
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   int&          error_code     // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-alatoires distribuées suivant la loi de distribution de Fisher avec les paramètres nu1 et nu2. En cas d'erreur, retourne false. Analogue à rf() dans R.

```
bool  MathRandomF(
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

nu1

[in] Le premier paramètre de distribution (nombre de degrés de liberté).

nu2

[in] Le deuxième paramètre de distribution (nombre de degrés de liberté).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsF

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution de Fisher avec les paramètres nu1 et nu2

```
double  MathMomentsF(
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewn
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

nu1

[in] Le premier paramètre de distribution (nombre de degrés de liberté).

nu2

[in] Le deuxième paramètre de distribution (nombre de degrés de liberté).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution de Fisher non-centrale

Cette section contient les fonctions permettant de travailler avec une distribution de Fisher non-centrale. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi de Fisher non-centrale. La distribution de Fisher non-centrale est définie par la formule suivante :

$$f_{NoncentralF}(x|\nu_1,\nu_2,\sigma)=e^{-\frac{\sigma}{2}}\sum_{r=0}^{\infty}\frac{1}{r!}\left(\frac{\sigma}{2}\right)^r\frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}+r\right)}{\Gamma\left(\frac{\nu_2}{2}+r\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_2}{2}+r}\frac{x^{\frac{\nu_2}{2}-1+r}}{\left(1+\frac{\nu_1}{\nu_2}x\right)^{\frac{\nu_1+\nu_2}{2}+r}}$$

Avec :

- x — valeur de la variable aléatoire
- ν1 — le premier paramètre de distribution (nombre de degrés de liberté)
- ν2 — le deuxième paramètre de distribution (nombre de degrés de liberté)
- σ — paramètre de noncentralité

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityNoncentralF	Calcule la fonction de densité de probabilité de la distribution de Fisher non-centrale
MathCumulativeDistributionNoncentralF	Calcule la valeur de la fonction de la distribution de Fisher non-centrale
MathQuantileNoncentralF	Calcule la valeur de la fonction inverse de distribution de Fisher non-centrale pour la probabilité spécifiée
MathRandomNoncentralF	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de distribution de Fisher non centrale
MathMomentsNoncentralF	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution de Fisher non-centrale

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNoncentralF

Calcule la fonction de densité de probabilité de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,             // valeur de la variable aléatoire
   const double  nu1,           // le premier paramètre de distribution (nombre de deg
   const double  nu2,           // le deuxième paramètre de distribution (nombre de de
   const double  sigma,         // paramètre de noncentralité
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,             // valeur de la variable aléatoire
   const double  nu1,           // le premier paramètre de distribution (nombre de deg
   const double  nu2,           // le deuxième paramètre de distribution (nombre de de
   const double  sigma,         // paramètre de noncentralité
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à df() dans R.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const double  sigma,          // paramètre de noncentralité
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const double  sigma,          // paramètre de noncentralité
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Paramètres

© 2000-2024, MetaQuotes Ltd.

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

nu1

[in] Le premier paramètre de distribution (nombre de degrés de liberté).

nu2

[in] Le deuxième paramètre de distribution (nombre de degrés de liberté).

sigma

[in] Paramètre de noncentralité.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralF

Calcule la fonction de densité de probabilité de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,             // valeur de la variable aléatoire
   const double  nu1,           // le premier paramètre de distribution (nombre de deg
   const double  nu2,           // le deuxième paramètre de distribution (nombre de de
   const double  sigma,         // paramètre de noncentralité
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,             // valeur de la variable aléatoire
   const double  nu1,           // le premier paramètre de distribution (nombre de deg
   const double  nu2,           // le deuxième paramètre de distribution (nombre de de
   const double  sigma,         // paramètre de noncentralité
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pf() dans R.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const double  sigma,          // paramètre de noncentralité
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution des probabilités de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const double  sigma,          // paramètre de noncentralité
   double&       result[]        // tableau de valeurs de la fonction de probabilité
```

© 2000-2024, MetaQuotes Ltd.

```
);
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

nu1

[in] Le premier paramètre de distribution (nombre de degrés de liberté).

nu2

[in] Le deuxième paramètre de distribution (nombre de degrés de liberté).

sigma

[in] Paramètre de noncentralité.

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNoncentralF

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma. En cas d'erreur, retourne NaN.

```
double  MathQuantileNoncentralF(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const double  sigma,          // paramètre de noncentralité
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma. En cas d'erreur, retourne NaN.

```
double  MathQuantileNoncentralF(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const double  sigma,          // paramètre de noncentralité
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution de Fisher non centrale inverse avec les paramètres nu1, nu2 et sigma. En cas d'erreur, retourne false. Analogue à qf() dans R.

```
double  MathQuantileNoncentralF(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const double  sigma,          // paramètre de noncentralité
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution de Fisher non centrale inverse avec les paramètres nu1, nu2 et sigma. En cas d'erreur, retourne false.

```
bool  MathQuantileNoncentralF(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   double&       result[]        // tableau des valeurs des quantiles
   );
```

© 2000-2024, MetaQuotes Ltd.

Paramètres

probability

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

nu1

[in] Le premier paramètre de distribution (nombre de degrés de liberté).

nu2

[in] Le deuxième paramètre de distribution (nombre de degrés de liberté).

sigma

[in] Paramètre de noncentralité.

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomNoncentralF

Génère une variable pseudo-alatoire distribuée suivant la loi de distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma. En cas d'erreur, retourne NaN.

```
double  MathRandomNoncentralF(
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const double  sigma,          // paramètre de noncentralité
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-alatoires distribuées suivant la loi de distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma. En cas d'erreur, retourne false. Analogue à rf() dans R.

```
bool  MathRandomNoncentralF(
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const double  sigma,          // paramètre de noncentralité
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

nu1

[in] Le premier paramètre de distribution (nombre de degrés de liberté).

nu2

[in] Le deuxième paramètre de distribution (nombre de degrés de liberté).

sigma

[in] Paramètre de noncentralité.

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNoncentralF

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution de Fisher non-centrale avec les paramètres nu1, nu2 et sigma.

```
double  MathMomentsNoncentralF(
   const double  nu1,            // le premier paramètre de distribution (nombre de de
   const double  nu2,            // le deuxième paramètre de distribution (nombre de d
   const double  sigma,          // paramètre de noncentralité
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewn
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

nu1

[in] Le premier paramètre de distribution (nombre de degrés de liberté).

nu2

[in] Le deuxième paramètre de distribution (nombre de degrés de liberté).

sigma

[in] Paramètre de noncentralité.

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution t

Cette section contient les fonctions permettant de travailler avec une distribution t. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi t (Student's t law). La distribution t est définie par la formule suivante :

$$f_T(x|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\pi\nu}} \frac{1}{\left(1+\frac{x^2}{\nu}\right)^{\frac{\nu+1}{2}}}$$

Avec :

- x — valeur de la variable aléatoire
- ν — paramètre de distribution (nombre de degrés de liberté)

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityT	Calcule la fonction de densité de probabilité de la distribution t
MathCumulativeDistributionT	Calcule la valeur de la fonction de la distribution t
MathQuantileT	Calcule la valeur de la fonction inverse de distribution t pour la probabilité spécifiée
MathRandomT	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de distribution t
MathMomentsT	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution t

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityT

Calcule la fonction de densité de probabilité de la distribution t avec le paramètre nu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityT(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution t avec le paramètre nu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityT(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution t avec le paramètre nu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dt() dans R.

```
bool  MathProbabilityDensityT(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Calcule la fonction de densité de probabilité de la distribution t avec le paramètre nu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityT(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

log_mode

© 2000-2024, MetaQuotes Ltd.

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionT

Calcule la valeur de la fonction de distribution t avec le paramètre nu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionT(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution t avec le paramètre nu pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionT(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution t avec le paramètre nu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pt() dans R.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la valeur de la fonction de distribution t avec le paramètre nu pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileT

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de distribution t inverse avec le paramètre nu. En cas d'erreur, retourne NaN.

```
double  MathQuantileT(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de distribution t inverse avec le paramètre nu. En cas d'erreur, retourne NaN.

```
double  MathQuantileT(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution t inverse avec le paramètre nu. En cas d'erreur, retourne false. Analogue à qt() dans R.

```
double  MathQuantileT(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution t inverse avec le paramètre nu. En cas d'erreur, retourne false.

```
bool  MathQuantileT(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

nu

© 2000-2024, MetaQuotes Ltd.

[in] Paramètre de distribution (nombre de degrés de liberté).

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomT

Génère une variable pseudo-alatoire distribuée suivant la loi de distribution t avec le paramètre nu. En cas d'erreur, retourne NaN.

```
double  MathRandomT(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-alatoires distribuées suivant la loi de distribution t avec le paramètre nu. En cas d'erreur, retourne false. Analogue à rt() dans R.

```
bool  MathRandomT(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsT

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution t avec le paramètre nu

```
double  MathMomentsT(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewn
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution t non-centrale

Cette section contient les fonctions permettant de travailler avec une distribution t non centrale. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi de distribution t non centrale. La distribution t non-centrale est définie par la formule suivante :

$$f_{NoncentralT}(x \mid \nu, \delta) = \frac{\nu^{\frac{\nu}{2}} e^{-\frac{\sigma^2}{2}}}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi}\left(\nu + x^2\right)^{\frac{\nu+1}{2}}} \sum_{r=0}^{\infty} \frac{(x\delta)^r}{r!} \left(\frac{2}{\nu + x^2}\right)^{\frac{r}{2}} \Gamma\left(\frac{\nu + r + 1}{2}\right)$$

Avec :

- x — valeur de la variable aléatoire
- ν — paramètre de distribution (nombre de degrés de liberté)
- σ — paramètre de noncentralité

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityNoncentralT	Calcule la fonction de densité de probabilité de la distribution t non-centrale
MathCumulativeDistributionNoncentralT	Calcule la valeur de la fonction de la distribution t non-centrale
MathQuantileNoncentralT	Calcule la valeur de la fonction inverse de distribution t non-centrale pour la probabilité spécifiée
MathRandomNoncentralT	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de distribution t non-centrale
MathMomentsNoncentralT	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution t non-centrale

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNoncentralT

Calcule la fonction de densité de probabilité de la distribution t non-centrale avec les paramètres nu et delta pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,         // paramètre de noncentralité
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=tr
   int&          error_code     // variable pour le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution t non-centrale avec les paramètres nu et delta pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,         // paramètre de noncentralité
   int&          error_code     // variable pour le code d'erreur
   );
```

Calcule la fonction de densité de probabilité de la distribution t non-centrale avec les paramètres nu et delta pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dt() dans R.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Calcule la fonction de densité de probabilité de la distribution t non-centrale avec les paramètres nu et delta pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   double&       result[]        // tableau de valeurs de la fonction de densité de pr
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

delta

[in] Paramètre de noncentralité.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralT

Calcule la fonction de distribution de probabilité de la distribution t non-centrale avec les paramètres nu et delta pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,         // paramètre de noncentralité
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution t non-centrale avec les paramètres nu et delta pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,             // valeur de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,         // paramètre de noncentralité
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution t non-centrale avec les paramètres nu et delta pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pt() dans R.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution de probabilité de la distribution t non-centrale avec les paramètres nu et delta pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

x[]

[in] Tableau des valeurs de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

delta

[in] Paramètre de noncentralité.

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNoncentralT

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de distribution t non-centrale inverse avec les paramètres nu et delta. En cas d'erreur, retourne NaN.

```
double  MathQuantileNoncentralT(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   const bool    tail,           // flag de calcul, si lower_tail=false, alors le calc
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de distribution t non-centrale inverse avec les paramètres nu et delta. En cas d'erreur, retourne NaN.

```
double  MathQuantileNoncentralT(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution t non centrale inverse avec les paramètres nu et delta. En cas d'erreur, retourne false. Analogue à qt() dans R.

```
double  MathQuantileNoncentralT(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   const bool    tail,           // flag de calcul, si lower_tail=false, alors le calc
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution t non centrale inverse avec les paramètres nu et delta. En cas d'erreur, retourne false.

```
bool  MathQuantileNoncentralT(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

© 2000-2024, MetaQuotes Ltd.

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

delta

[in] Paramètre de noncentralité.

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomNoncentralT

Génère une variable pseudo-alatoire distribuée suivant la loi de distribution t non-centrale avec les paramètres nu et delta. En cas d'erreur, retourne NaN.

```
double  MathRandomNoncentralT(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-alatoires distribuées suivant la loi de distribution t non-centrale avec les paramètres nu et delta. En cas d'erreur, retourne false. Analogue à rt() dans R.

```
bool  MathRandomNoncentralT(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

delta

[in] Paramètre de noncentralité.

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNoncentralT

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution t non-centrale avec les paramètres nu et delta.

```
double  MathMomentsNoncentralT(
   const double  nu,            // paramètre de la distribution (nombre de degrés de l
   const double  delta,          // paramètre de noncentralité
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewn
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

nu

[in] Paramètre de distribution (nombre de degrés de liberté).

delta

[in] Paramètre de noncentralité.

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution logistique

Cette section contient les fonctions utilisées pour travailler avec une distribution logistique. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi logistque. La distribution logistique est définie par la formule suivante :

$$f_{Logistic}(x \mid \mu, \sigma) = \frac{e^{\frac{x-\mu}{\sigma}}}{\sigma\left(1+e^{\frac{x-\mu}{\sigma}}\right)^2}$$

Avec :

- x — valeur de la variable aléatoire
- μ — paramètre moyen de la distribution
- σ — paramètre d'échelle de la distribution

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityLogistic	Calcule la fonction de densité de probabilité de la distribution logistique
MathCumulativeDistributionLogistic	Calcule la valeur de la fonction de distribution de probabilité logistique
MathQuantileLogistic	Calcule la valeur de la fonction inverse de distribution logistique pour la probabilité spécifiée
MathRandomLogistic	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi logistique
MathMomentsLogistic	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution logistique

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityLogistic

Calcule la valeur de la fonction de densité de probabilité de la distribution logistique avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityLogistic(
   const double  x,             // valeur de la variable aléatoire
   const double  mu,            // paramètre moyen de la distribution
   const double  sigma,         // paramètre d'échelle de la distribution
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=t
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution logistique avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityLogistic(
   const double  x,             // valeur de la variable aléatoire
   const double  mu,            // paramètre moyen de la distribution
   const double  sigma,         // paramètre d'échelle de la distribution
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité des probabilités de la distribution logistique avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dlogis() dans R.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de densité des probabilités de la distribution logistique avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

mu

[in] paramètre moyen de la distribution.

sigma

[in] paramètre d'échelle de la distribution.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionLogistic

Calcule la fonction de distribution logistique des probabilités avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionLogistic(
   const double  x,             // valeur de la variable aléatoire
   const double  mu,            // paramètre moyen de la distribution
   const double  sigma,         // paramètre d'échelle de la distribution
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution logistique des probabilités avec les paramètres mu et sigma pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionLogistic(
   const double  x,             // valeur de la variable aléatoire
   const double  mu,            // paramètre moyen de la distribution
   const double  sigma,         // paramètre d'échelle de la distribution
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution logistique des probabilités avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution logistique des probabilités avec les paramètres mu et sigma pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à plogis() dans R.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

mu

[in] paramètre moyen de la distribution.

sigma

[in] paramètre d'échelle de la distribution.

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileLogistic

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de distribution logistique inverse avec les paramètres mu et sigma. En cas d'erreur, retourne NaN.

```
double  MathQuantileLogistic(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction de distribution logistique inverse avec les paramètres mu et sigma. En cas d'erreur, retourne NaN.

```
double  MathQuantileLogistic(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution logistique inverse avec les paramètres mu et sigma. En cas d'erreur, retourne false. Analogue à qlogis() dans R.

```
double  MathQuantileLogistic(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution logistique inverse avec les paramètres mu et sigma. En cas d'erreur, retourne false.

```
bool  MathQuantileLogistic(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

© 2000-2024, MetaQuotes Ltd.

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

mu

[in] paramètre moyen de la distribution.

sigma

[in] paramètre d'échelle de la distribution.

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomLogistic

Génère une variable pseudo-aléatoire distribuée suivant la loi de la distribution logistique avec les paramètres mu et sigma. En cas d'erreur, retourne NaN.

```
double  MathRandomLogistic(
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi de la distribution logistique avec les paramètres mu et sigma. En cas d'erreur, retourne false. Analogue à rlogis() dans R.

```
bool  MathRandomLogistic(
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

mu

[in] paramètre moyen de la distribution.

sigma

[in] paramètre d'échelle de la distribution.

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsLogistic

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution logistique avec les paramètres mu et sigma.

```
double  MathMomentsLogistic(
   const double  mu,             // paramètre moyen de la distribution
   const double  sigma,          // paramètre d'échelle de la distribution
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

mu

[in] paramètre moyen de la distribution.

sigma

[in] paramètre d'échelle de la distribution.

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution de Cauchy

Cette section contient les fonctions permettant de travailler avec une distribution de Cauchy. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi de Cauchy. La distribution de Cauchy est définie par la formule suivante :

$$f_{Cauchy}(x|a,b) = \frac{1}{\pi}\frac{b}{\left(b^2+(x-a)^2\right)}$$

Avec :

- x — valeur de la variable aléatoire
- a — paramètre moyen de la distribution
- b — paramètre d'échelle de la distribution

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityCauchy	Calcule la fonction de densité de probabilité de la distribution de Cauchy
MathCumulativeDistributionCauchy	Calcule la valeur de la fonction de distribution de probabilité de Cauchy
MathQuantileCauchy	Calcule la valeur de la fonction inverse de distribution de Cauchy pour la probabilité spécifiée
MathRandomCauchy	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de Cauchy
MathMomentsCauchy	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution de Cauchy

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityCauchy

Calcule la valeur de la fonction de densité de probabilité de la distribution de Cauchy avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityCauchy(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre moyen de la distribution
   const double  b,             // paramètre d'échelle de la distribution
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=t
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution de Cauchy avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityCauchy(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre moyen de la distribution
   const double  b,             // paramètre d'échelle de la distribution
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution de Cauchy avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dcauchy() dans R.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution de Cauchy avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

a

[in] paramètre moyen de la distribution.

b

[in] paramètre d'échelle de la distribution.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionCauchy

Calcule la fonction de distribution de probabilité de la distribution de Cauchy avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionCauchy(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre moyen de la distribution
   const double  b,             // paramètre d'échelle de la distribution
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution de Cauchy avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionCauchy(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre moyen de la distribution
   const double  b,             // paramètre d'échelle de la distribution
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution de Cauchy avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution de probabilité de la distribution de Cauchy avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pcauchy() dans R.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

x[]

[in] Tableau des valeurs de la variable aléatoire.

a

[in] paramètre moyen de la distribution.

b

[in] paramètre d'échelle de la distribution.

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileCauchy

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction inverse de distribution de Cauchy avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathQuantileCauchy(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction inverse de distribution de Cauchy avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathQuantileCauchy(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution de Cauchy inverse avec les paramètres a et b En cas d'erreur, retourne false. Analogue à qcauschy() dans R.

```
double  MathQuantileCauchy(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution de Cauchy inverse avec les paramètres a et b En cas d'erreur, retourne false.

```
bool  MathQuantileCauchy(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

a

[in] paramètre moyen de la distribution.

b

[in] paramètre d'échelle de la distribution.

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomCauchy

Génère une variable pseudo-aléatoire distribuée suivant la loi de la distribution de Cauchy avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathRandomCauchy(
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi de la distribution de Cauchy avec les paramètres a et b. En cas d'erreur, retourne false. Analogue à rcauchy() dans R.

```
bool  MathRandomCauchy(
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

a

[in] paramètre moyen de la distribution.

b

[in] paramètre d'échelle de la distribution.

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsCauchy

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution de Cauchy avec les paramètres a et b.

```
double  MathMomentsCauchy(
   const double  a,              // paramètre moyen de la distribution
   const double  b,              // paramètre d'échelle de la distribution
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

a

[in] paramètre moyen de la distribution.

b

[in] paramètre d'échelle de la distribution.

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution uniforme

Cette section contient les fonctions permettant de travailler avec une distribution uniforme. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi uniforme. La distribution uniforme est définie par la formule suivante :

$$f_{Uniform}(x|\,a,b)=\frac{1}{b-a}$$

Avec :

- x — valeur de la variable aléatoire
- a — paramètre de la distribution (limite inférieure)
- b — paramètre de la distribution (limite supérieure)

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityUniform	Calcule la fonction de densité de probabilité de la distribution uniforme
MathCumulativeDistributionUniform	Calcule la valeur de la fonction de distribution de probabilité uniforme
MathQuantileUniform	Calcule la valeur de la fonction inverse de distribution uniforme pour la probabilité spécifiée
MathRandomUniform	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi uniforme
MathMomentsUniform	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution uniforme

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityUniform

Calcule la valeur de la fonction de densité de probabilité de la distribution uniforme avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityUniform(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=t
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution uniforme avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityUniform(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution uniforme avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dunif() dans R.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution uniforme avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

a

[in] Paramètre a de la distribution (limite inférieure).

b

[in] Paramètre b de la distribution (limite supérieure).

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionUniform

Calcule la fonction de distribution de probabilité de la distribution uniforme avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionUniform(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution uniforme avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionUniform(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la fonction de distribution de probabilité de la distribution uniforme avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la fonction de distribution de probabilité de la distribution uniforme avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à punif() dans R.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

a

[in] Paramètre a de la distribution (limite inférieure).

b

[in] Paramètre b de la distribution (limite supérieure).

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileUniform

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction inverse de distribution uniforme avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathQuantileUniform(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction inverse de distribution uniforme avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathQuantileUniform(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution uniforme inverse avec les paramètres a et b. En cas d'erreur, retourne false. Analogue à qunif() dans R.

```
double  MathQuantileUniform(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution uniforme inverse avec les paramètres a et b. En cas d'erreur, retourne false.

```
bool  MathQuantileUniform(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

a

[in] Paramètre a de la distribution (limite inférieure).

b

[in] Paramètre b de la distribution (limite supérieure).

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomUniform

Génère une variable pseudo-aléatoire distribuée suivant la loi de la distribution uniforme avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathRandomUniform(
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi de la distribution uniforme avec les paramètres a et b. En cas d'erreur, retourne false. Analogue à runif() dans R.

```
bool  MathRandomUniform(
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

a

[in] Paramètre a de la distribution (limite inférieure).

b

[in] Paramètre b de la distribution (limite supérieure).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsUniform

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution uniforme avec les paramètres a et b.

```
double  MathMomentsUniform(
   const double  a,             // paramètre a de la distribution (limite inférieure)
   const double  b,             // paramètre b de la distribution (limite supérieure)
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

a

[in] Paramètre a de la distribution (limite inférieure).

b

[in] Paramètre b de la distribution (limite supérieure).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution Weibull

Cette section contient les fonctions utilisées pour travailler avec une distribution Weibull. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi de Weibull. La distribution de Weibull est définie par la formule suivante :

$$f_{Weibull}(x|a,b) = \frac{a}{b}\left(\frac{x}{b}\right)^{a-1} e^{-\left(\frac{x}{b}\right)^{a}}$$

Avec :

- x — valeur de la variable aléatoire
- a — paramètre de la distribution (forme)
- b — paramètre de la distribution (échelle)

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityWeibull	Calcule la fonction de densité de probabilité de la distribution de Weibull
MathCumulativeDistributionWeibull	Calcule la valeur de la fonction de distribution de probabilité de Weibull
MathQuantileWeibull	Calcule la valeur de la fonction inverse de distribution de Weibull pour la probabilité spécifiée
MathRandomWeibull	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de Weibull
MathMomentsWeibull	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution de Weibull

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityWeibull

Calcule la valeur de la fonction de densité de probabilité de la distribution de Weibull avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityWeibull(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre de la distribution (forme)
   const double  b,             // paramètre de la distribution (échelle)
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=t
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution de Weibull avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityWeibull(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre de la distribution (forme)
   const double  b,             // paramètre de la distribution (échelle)
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution de Weibull avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dweibull() dans R.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de densité de probabilité de la distribution de Weibull avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

a

[in] Paramètre de la distribution (échelle).

b

[in] Paramètre de la distribution (forme).

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionWeibull

Calcule la valeur de la fonction de distribution de Weibull avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionWeibull(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre de la distribution (forme)
   const double  b,             // paramètre de la distribution (échelle)
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution de Weibull avec les paramètres a et b pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionWeibull(
   const double  x,             // valeur de la variable aléatoire
   const double  a,             // paramètre de la distribution (forme)
   const double  b,             // paramètre de la distribution (échelle)
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution de Weibull avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pweibull() dans R.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la valeur de la fonction de distribution de Weibull avec les paramètres a et b pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

a

[in] Paramètre de la distribution (échelle).

b

[in] Paramètre de la distribution (forme).

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileWeibull

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction inverse de distribution de Weibull avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathQuantileWeibull(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur de la fonction inverse de distribution de Weibull avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathQuantileWeibull(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution de Weibull inverse avec les paramètres a et b. En cas d'erreur, retourne false. Analogue à qweibull() dans R.

```
double  MathQuantileWeibull(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur de la fonction de distribution de Weibull inverse avec les paramètres a et b. En cas d'erreur, retourne false.

```
bool  MathQuantileWeibull(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

a

[in] Paramètre de la distribution (échelle).

b

[in] Paramètre de la distribution (forme).

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomWeibull

Génère une variable pseudo-aléatoire distribuée suivant la loi de la distribution de Weibull avec les paramètres a et b. En cas d'erreur, retourne NaN.

```
double  MathRandomWeibull(
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi de la distribution de Weibull avec les paramètres a et b. En cas d'erreur, retourne false. Analogue à rweibull() dans R.

```
bool  MathRandomWeibull(
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

a

[in] Paramètre de la distribution (échelle).

b

[in] Paramètre de la distribution (forme).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsWeibull

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution de Weibull avec les paramètres a et b.

```
double  MathMomentsWeibull(
   const double  a,              // paramètre de la distribution (forme)
   const double  b,              // paramètre de la distribution (échelle)
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

a

[in] Paramètre de la distribution (échelle).

b

[in] Paramètre de la distribution (forme).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution binomiale

Cette section contient les fonctions permettant de travailler avec une distribution binomiale. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi binomiale. La distribution binomiale est définie par la formule suivante :

$$f_{Binomial}(x \mid n, p) = \binom{n}{x} p^x (1-p)^{n-x}$$

Avec :

- x — valeur de la variable aléatoire
- n — nombre de tests
- p — probabilité de succès de chaque test

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityBinomial	Calcule la fonction de densité de probabilité de la distribution binomiale
MathCumulativeDistributionBinomial	Calcule la valeur de la fonction de distribution de probabilité binomiale
MathQuantileBinomial	Calcule la valeur de la fonction inverse de distribution binomiale pour la probabilité spécifiée
MathRandomBinomial	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi binomiale
MathMomentsBinomial	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution binomiale

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityBinomial

Calcule la valeur de la fonction de masse de probabilité de la distribution binomiale avec les paramètres n et p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // valeur de la variable aléatoire
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occur
   const bool    log_mode,        // calcule le logarithme de la valeur, si log_mode=t
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution binomiale avec les paramètres n et p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // valeur de la variable aléatoire
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occur
   int&          error_code      // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution binomiale avec les paramètres n et p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dbinom() dans R.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution binomiale avec les paramètres n et p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

n

[in] Paramètre de la distribution (nombre de tests).

p

[in] Paramètre de la distribution (probabilité ou occurrence de l'évènement dans un test).

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionBinomial

Calcule la valeur de la fonction de distribution de probabilité pour la loi binomiale avec les paramètres n et p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionBinomial(
   const double  x,             // valeur de la variable aléatoire
   const double  n,             // paramètre de la distribution (nombre de tests)
   const double  p,             // paramètre de la distribution (probabilité ou occur
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution de probabilité pour la loi binomiale avec les paramètres n et p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionBinomial(
   const double  x,             // valeur de la variable aléatoire
   const double  n,             // paramètre de la distribution (nombre de tests)
   const double  p,             // paramètre de la distribution (probabilité ou occur
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution de probabilité pour la loi binomiale avec les paramètres n et p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pbinom() dans R.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la valeur de la fonction de distribution de probabilité pour la loi binomiale avec les paramètres n et p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

x[]

[in] Tableau des valeurs de la variable aléatoire.

n

[in] Paramètre de la distribution (nombre de tests).

p

[in] Paramètre de la distribution (probabilité ou occurrence de l'évènement dans un test).

tail

[in] Flag de calcul. Si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileBinomial

Pour la *probabilité* spécifiée, la fonction calcule la valeur inverse de la fonction de distribution pour la loi binomiale avec les paramètres n et p. En cas d'erreur, retourne NaN.

```
double  MathQuantileBinomial(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est e
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur inverse de la fonction de distribution pour la loi binomiale avec les paramètres n et p. En cas d'erreur, retourne NaN.

```
double  MathQuantileBinomial(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur inverse de la fonction de distribution pour la loi binomiale avec les paramètres n et p En cas d'erreur, retourne false. Analogue à qbinom() dans R.

```
double  MathQuantileBinomial(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est e
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur inverse de la fonction de distribution pour la loi binomiale avec les paramètres n et p En cas d'erreur, retourne false.

```
bool  MathQuantileBinomial(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

© 2000-2024, MetaQuotes Ltd.

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

n

[in] Paramètre de la distribution (nombre de tests).

p

[in] Paramètre de la distribution (probabilité ou occurrence de l'évènement dans un test).

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomBinomial

Génère une variable pseudo-aléatoire distribuée suivant la loi de la distribution normale avec les paramètres n et p. En cas d'erreur, retourne NaN.

```
double  MathRandomBinomial(
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi de la distribution binomiale avec les paramètres n et p. En cas d'erreur, retourne false. Analogue à rbinom() dans R.

```
bool  MathRandomBinomial(
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

n

[in] Paramètre de la distribution (nombre de tests).

p

[in] Paramètre de la distribution (probabilité ou occurrence de l'évènement dans un test).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsBinomial

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution binomiale avec les paramètres n et p.

```
double  MathMomentsBinomial(
   const double  n,              // paramètre de la distribution (nombre de tests)
   const double  p,              // paramètre de la distribution (probabilité ou occu
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skew
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

n

[in] Paramètre de la distribution (nombre de tests).

p

[in] Paramètre de la distribution (probabilité ou occurrence de l'évènement dans un test).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution binomiale négative

Cette section contient les fonctions permettant de travailler avec une distribution binomiale négative. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi binomiale négative. La distribution binomiale négative est définie par la formule suivante :

$$f_{NegativeBinomial}(x \mid r, p) = \frac{\Gamma(r+x)}{\Gamma(r)\Gamma(x+1)} p^r (1-p)^x$$

Avec :

- x — valeur de la variable aléatoire
- r — nombre de tests réussis
- p — probabilité de succès

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityNegativeBinomial	Calcule la fonction de densité de probabilité de la distribution binomiale négative
MathCumulativeDistributionNegativeBinomial	Calcule la valeur de la fonction de distribution de probabilité binomiale négative
MathQuantileNegativeBinomial	Calcule la valeur de la fonction inverse de distribution binomiale négative pour la probabilité spécifiée
MathRandomNegativeBinomial	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi binomiale négative
MathMomentsNegativeBinomial	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution binomiale négative

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNegativeBinomial

Calcule la valeur de la fonction de masse de probabilité de la distribution binomiale négative avec les paramètres r et p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  r,             // nombre de tests réussis
   const double  p,             // probabilité de succès
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=t
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution binomiale négative avec les paramètres r et p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  r,             // nombre de tests réussis
   const double  p,             // probabilité de succès
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution binomiale négative avec les paramètres r et p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dnbinom() dans R.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution binomiale négative avec les paramètres r et p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Tableau des valeurs de la variable aléatoire.

r

[in] Nombre de tests réussis

p

[in] Probabilité de succès.

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNegativeBinomial

Calcule la valeur de la foncition de distribution des probabilités pour la loi binomiale négative avec les paramètres r et p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  r,             // nombre de tests réussis
   const double  p,             // probabilité de succès
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la foncition de distribution des probabilités pour la loi binomiale négative avec les paramètres r et p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  r,             // nombre de tests réussis
   const double  p,             // probabilité de succès
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution de probabilités pour la loi binomiale négative avec les paramètres r et p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pweibull() dans R.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la valeur de la fonction de distribution de probabilités pour la loi binomiale négative avec les paramètres r et p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

© 2000-2024, MetaQuotes Ltd.

x[]

[in] Tableau des valeurs de la variable aléatoire.

r

[in] Nombre de tests réussis.

p

[in] Probabilité de succès.

tail

[in] Flag de calcul, si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur, si log_mode=true, alors le logarithme naturel de la probabilité est calculée.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNegativeBinomial

Pour la *probabilité* spécifiée, la fonction calcule la valeur inverse de la fonction de distribution pour la loi binomiale négative avec les paramètres r et p. En cas d'erreur, retourne NaN.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur inverse de la fonction de distribution pour la loi binomiale négative avec les paramètres r et p. En cas d'erreur, retourne NaN.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur inverse de la fonction de distribution pour la loi binomiale négative avec les paramètres r et p En cas d'erreur, retourne false. Analogue à qnbinom() dans R.

```
double  MathQuantileNegativeBinomial(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur inverse de la fonction de distribution pour la loi binomiale négative avec les paramètres r et p En cas d'erreur, retourne false.

```
bool  MathQuantileNegativeBinomial(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

© 2000-2024, MetaQuotes Ltd.

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

r

[in] Nombre de tests réussis.

p

[in] Probabilité de succès.

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomNegativeBinomial

Génère une variable pseudo-aléatoire distribuée suivant la loi de la distribution binomiale négative avec les paramètres r et p. En cas d'erreur, retourne NaN.

```
double  MathRandomNegativeBinomial(
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi de distribution binomiale négative avec les paramètres r et p. En cas d'erreur, retourne false. Analogue à rweibull() dans R.

```
bool  MathRandomNegativeBinomial(
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

r

[in] Nombre de tests réussis.

p

[in] Probabilité de succès.

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNegativeBinomial

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution binomiale négative avec les paramètres r et p.

```
double  MathMomentsNegativeBinomial(
   const double  r,              // nombre de tests réussis
   const double  p,              // probabilité de succès
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

r

[in] Nombre de tests réussis.

p

[in] Probabilité de succès.

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution géométrique

Cette section contient les fonctions permettant de travailler avec une distribution géométrique. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi géométrique. La distribution géométrique est définie par la formule suivante :

$$f_{Geometric}(x|p) = p(1-p)^x$$

Avec :

- x — valeur de la variable aléatoire (integer)
- p — probabilité de survenance d'un évènement dans un test

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityGeometric	Calcule la fonction de densité de probabilité de la distribution géométrique
MathCumulativeDistributionGeometric	Calcule la valeur de la fonction de distribution de probabilité géométrique
MathQuantileGeometric	Calcule la valeur de la fonction de distribution géométrique inverse pour la probabilité spécifiée
MathRandomGeometric	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi géométrique
MathMomentsGeometric	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution géométrique

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityGeometric

Calcule la valeur de la fonction de masse de probabilité de la distribution géométrique avec le paramètre p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityGeometric(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  p,             // paramètre de la distribution (probabilité ou occur
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=t
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution géométrique avec le paramètre p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityGeometric(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  p,             // paramètre de la distribution (probabilité ou occur
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution géométrique avec le paramètre p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dgeom() dans R.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  p,              // paramètre de la distribution (probabilité ou occu
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution géométrique avec le paramètre p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  p,              // paramètre de la distribution (probabilité ou occu
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

p

[in] Paramètre de la distribution (probabilité ou occurrence de l'évènement dans un test).

log_mode

© 2000-2024, MetaQuotes Ltd.

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionGeometric

Calcule la valeur de la fonction de distribution de probabilité pour la loi géométrique avec le paramètre p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionGeometric(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  p,             // paramètre de la distribution (probabilité ou occur
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution de probabilité pour la loi géométrique avec le paramètre p pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionGeometric(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  p,             // paramètre de la distribution (probabilité ou occur
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution de probabilité pour la loi géométrique avec le paramètre p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à pgeom() dans R.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  p,              // paramètre de la distribution (probabilité ou occu
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Calcule la valeur de la fonction de distribution de probabilité pour la loi géométrique avec le paramètre p pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  p,              // paramètre de la distribution (probabilité ou occu
   double&       result[]        // tableau de valeurs de la fonction de probabilité
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

p

© 2000-2024, MetaQuotes Ltd.

[in] Paramètre de la distribution (probabilité ou occurrence de l'évènement dans un test).

tail

[in] Flag de calcul, si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur, si log_mode=true, alors le logarithme naturel de la probabilité est calculée.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathQuantileGeometric

Pour la *probabilité* spécifiée, la fonction calcule la valeur inverse de la fonction de distribution pour la loi géométrique avec le paramètre p. En cas d'erreur, retourne NaN.

```
double  MathQuantileGeometric(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  p,              // paramètre de la distribution (probabilité ou occu
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est e
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur inverse de la fonction de distribution pour la loi géométrique avec le paramètre p. En cas d'erreur, retourne NaN.

```
double  MathQuantileGeometric(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  p,              // paramètre de la distribution (probabilité ou occu
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur inverse de la fonction de distribution de la loi géométrique avec le paramètre p En cas d'erreur, retourne false. Analogue à qgeom() dans R.

```
double  MathQuantileGeometric(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  p,              // paramètre de la distribution (probabilité ou occu
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est e
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur inverse de la fonction de distribution de la loi géométrique avec le paramètre p En cas d'erreur, retourne false.

```
bool  MathQuantileGeometric(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  p,              // paramètre de la distribution (probabilité ou occu
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

p

© 2000-2024, MetaQuotes Ltd.

[in] Paramètre de la distribution (probabilité ou occurrence de l'évènement dans un test).

tail

[in] Flag de calcul, si false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomGeometric

Génère une variable pseudo-alatoire distribuée suivant la loi de distribution géométrique avec le paramètre p. En cas d'erreur, retourne NaN.

```
double  MathRandomGeometric(
   const double  p,              // paramètre de la distribution (probabilité ou occu
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-alatoires distribuées suivant la loi de distribution géométrique avec le paramètre p. En cas d'erreur, retourne false. Analogue à rgeom() dans R.

```
bool  MathRandomGeometric(
   const double  p,              // paramètre de la distribution (probabilité ou occu
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

p

[in] Paramètre de la distribution (probabilité ou occurrence de l'évènement dans un test).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsGeometric

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution géométrique avec le paramètre p.

```
double  MathMomentsGeometric(
   const double  p,              // paramètre de la distribution (probabilité ou occu
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skew
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

p

[in] Paramètre de la distribution (probabilité ou occurrence de l'évènement dans un test).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distribution Hypergéometrique

Cette section contient les fonctions utilisées pour travailler avec une distribution hyper-géométrique. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi hypergéometrique. La distribution hypergéometrique est définie selon la formule suivante :

$$f_{Hypergeometric}(x \mid m, k, n) = \frac{\binom{k}{x}\binom{m-k}{n-x}}{\binom{m}{n}}$$

Avec :

- x — valeur de la variable aléatoire (integer)
- m — nombre total d'objets
- k — nombre d'objets avec les caractéristiques désirées
- n — nombre de tirages d'objets

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityHypergeometric	Calcule la fonction de densité de probabilté de la distribution hypergéometrique
MathCumulativeDistributionHypergeometric	Calcule la valeur de la fonction de distribution de probabilité hypergéométrique
MathQuantileHypergeometric	Calcule la valeur de la fonction inverse de distribution hypergéométrique pour la probabilité spécifiéee
MathRandomHypergeometric	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi hypergéométrique
MathMomentsHypergeometric	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution hypergéométrique

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityHypergeometric

Calcule les valeurs de la fonction de masse de probabilité de distribution hypergéométrique avec les paramètres m, k et n pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  m,             // nombre total d'objets (integer)
   const double  k,             // nombre d'objets avec les caractéristiques désirées
   const double  n,             // nombre de tirages d'objets (integer)
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=t
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule les valeurs de la fonction de masse de probabilité de distribution hypergéométrique avec les paramètres m, k et n pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  m,             // nombre total d'objets (integer)
   const double  k,             // nombre d'objets avec les caractéristiques désirées
   const double  n,             // nombre de tirages d'objets (integer)
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule les valeurs de la fonction de masse de probabilité de distribution hypergéométrique avec les paramètres m, k et n pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dhyper() dans R.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule les valeurs de la fonction de masse de probabilité de distribution hypergéométrique avec les paramètres m, k et n pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

© 2000-2024, MetaQuotes Ltd.

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

m

[in] Nombre total d'objets (integer).

k

[in] Nombre d'objets avec les caractéristiques désirées (integer).

n

[in] Nombre de tirages d'objets (integer).

log_mode

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionHypergeometric

Calcule les valeurs de la fonction de distribution de probabilité pour la loi hypergéométrique avec les paramètres m, k et n, pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  m,             // nombre total d'objets (integer)
   const double  k,             // nombre d'objets avec les caractéristiques désirées
   const double  n,             // nombre de tirages d'objets (integer)
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule les valeurs de la fonction de distribution de probabilité pour la loi hypergéométrique avec les paramètres m, k et n, pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  m,             // nombre total d'objets (integer)
   const double  k,             // nombre d'objets avec les caractéristiques désirées
   const double  n,             // nombre de tirages d'objets (integer)
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule les valeurs de la fonction de distribution de probabilité pour la loi hypergéométrique avec les paramètres m, k et n, pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à phyper() dans R.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau des valeurs de la fonction de distributio
   );
```

Calcule les valeurs de la fonction de distribution de probabilité pour la loi hypergéométrique avec les paramètres m, k et n, pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   double&       result[]        // tableau des valeurs de la fonction de distributio
   );
```

© 2000-2024, MetaQuotes Ltd.

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

m

[in] Nombre total d'objets (integer).

k

[in] Nombre d'objets avec les caractéristiques désirées (integer).

n

[in] Nombre de tirages d'objets (integer).

tail

[in] Flag de calcul, si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur, si log_mode=true, alors le logarithme naturel de la probabilité est calculée.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau des valeurs de la fonction de distribution.

© 2000-2024, MetaQuotes Ltd.

MathQuantileHypergeometric

Pour la *probabilité* spécifiée, la fonction calcule la valeur inverse de la fonction de distribution pour la loi hypergéométrique avec les paramètres m, k et n. En cas d'erreur, retourne NaN.

```
double  MathQuantileHypergeometric(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est e
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur inverse de la fonction de distribution pour la loi hypergéométrique avec les paramètres m, k et n. En cas d'erreur, retourne NaN.

```
double  MathQuantileHypergeometric(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur inverse de la fonction de distribution de la loi hypergéométrique avec les paramètres m, k et n. En cas d'erreur, retourne false. Analogue à qhyper() dans R.

```
double  MathQuantileHypergeometric(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est e
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur inverse de la fonction de distribution de la loi hypergéométrique avec les paramètres m, k et n. En cas d'erreur, retourne false.

```
bool  MathQuantileHypergeometric(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   double&       result[]        // tableau des valeurs des quantiles
```

© 2000-2024, MetaQuotes Ltd.

```
);
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

m

[in] Nombre total d'objets (integer).

k

[in] Nombre d'objets avec les caractéristiques désirées (integer).

n

[in] Nombre de tirages d'objets (integer).

tail

[in] Flag de calcul, si tail=false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomHypergeometric

Génère une variable pseudo-aléatoire distribuée suivant la loi de la distribution hypergéométrique avec les paramètres m, n et k. En cas d'erreur, retourne NaN.

```
double  MathRandomHypergeometric(
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-aléatoires distribuées suivant la loi de la distribution hypergéométrique avec les paramètres m, n et k. En cas d'erreur, retourne false. Analogue à rhyper() dans R.

```
bool  MathRandomHypergeometric(
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   const int     data_count,     // quantité de données nécessaires
   double&       result[]        // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

m

[in] Nombre total d'objets (integer).

k

[in] Nombre d'objets avec les caractéristiques désirées (integer).

n

[in] Nombre de tirages d'objets (integer).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsHypergeometric

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution hypergéométrique avec les paramètres m, n et k.

```
double  MathMomentsHypergeometric(
   const double  m,              // nombre total d'objets (integer)
   const double  k,              // nombre d'objets avec les caractéristiques désirées
   const double  n,              // nombre de tirages d'objets (integer)
   double&       mean,           // variable pour la moyenne
   double&       variance,       // variable pour la variance
   double&       skewness,       // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,       // variable pour le kurtosis
   int&          error_code      // variable pour le code d'erreur
   );
```

Paramètres

m

[in] Nombre total d'objets (integer).

k

[in] Nombre d'objets avec les caractéristiques désirées (integer).

n

[in] Nombre de tirages d'objets (integer).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Distributio de Poisson

Cette section contient les fonctions permettant de travailler avec une distribution de Poisson. Elles permettent de calculer la densité, la probabilité, les quantiles et de générer des nombres pseudo-aléatoires distribués suivant la loi de Poisson. La distribution de Poisson est définie par la formule suivante :

$$f_{Poisson}(x|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}$$

Avec :

- x — valeur de la variable aléatoire
- λ — paramètre de la distribution (moyenne)

En complément du calcul de variables aléatoires individuelles, la bibliothèque implémente également la possibilité de travailler avec les tableaux de variables aléatoires.

Fonction	Description
MathProbabilityDensityPoisson	Calcule la fonction de densité de probabilité de la distribution de Poisson
MathCumulativeDistributionPoisson	Calcule la valeur de la fonction de distribution de probabilité de Poisson
MathQuantilePoisson	Calcule la valeur de la fonction inverse de distribution de Poisson pour la probabilité spécifiée
MathRandomPoisson	Génère une variable pseudo-aléatoire/un tableau de variables pseudo-aléatoires distribuées suivant la loi de Poisson
MathMomentsPoisson	Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution de Poisson

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityPoisson

Calcule la valeur de la fonction de masse de probabilité de la distribution de Poisson avec le paramètre lambda pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityPoisson(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  lambda,        // paramètre de la distribution (moyenne)
   const bool    log_mode,       // calcule le logarithme de la valeur, si log_mode=t
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution de Poisson avec le paramètre lambda pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathProbabilityDensityPoisson(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  lambda,        // paramètre de la distribution (moyenne)
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution de Poisson avec le paramètre lambda pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à dpois() dans R.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  lambda,         // paramètre de la distribution (moyenne)
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Calcule la valeur de la fonction de masse de probabilité de la distribution de Poisson avec le paramètre lambda pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  lambda,         // paramètre de la distribution (moyenne)
   double&       result[]        // tableau de valeurs de la fonction de densité de p
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

lambda

[in] Paramètre de la distribution (moyenne).

log_mode

© 2000-2024, MetaQuotes Ltd.

[in] Flag pour calculer le logarithme de la valeur. Si log_mode=true, alors le logarithme naturel de la densité de probabilité est calculé.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau de valeurs de la fonction de densité de probabilité.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionPoisson

Calcule la valeur de la fonction de distribution de Poisson avec le paramètre lambda pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionPoisson(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  lambda,        // paramètre de la distribution (moyenne)
   const bool    tail,          // flag de calcul, si true, alors la probabilité de la
   const bool    log_mode,      // flag pour calculer le logarithme de la valeur, si l
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution de Poisson avec le paramètre lambda pour une variable aléatoire x. En cas d'erreur, retourne NaN.

```
double  MathCumulativeDistributionPoisson(
   const double  x,             // valeur de la varialbe aléatoire (integer)
   const double  lambda,        // paramètre de la distribution (moyenne)
   int&          error_code     // variable stockant le code d'erreur
   );
```

Calcule la valeur de la fonction de distribution de Poisson avec le paramètre lambda pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false. Analogue à ppois() dans R.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  lambda,         // paramètre de la distribution (moyenne)
   const bool    tail,           // flag de calcul, si true, alors la probabilité que
   const bool    log_mode,       // flag pour calculer le logarithme de la valeur, si
   double&       result[]        // tableau des valeurs de la fonction de distributio
   );
```

Calcule la valeur de la fonction de distribution de Poisson avec le paramètre lambda pour un tableau de variables aléatoires x[]. En cas d'erreur, retourne false.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // tableau de valeurs de la variable aléatoire
   const double  lambda,         // paramètre de la distribution (moyenne)
   double&       result[]        // tableau des valeurs de la fonction de distributio
   );
```

Paramètres

x

[in] Valeur de la variable aléatoire.

x[]

[in] Tableau des valeurs de la variable aléatoire.

lambda

[in] Paramètre de la distribution (moyenne).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Flag de calcul, si true, alors la probabilité de la variable aléatoire ne dépassant pas x est calculée.

log_mode

[in] Flag pour calculer le logarithme de la valeur, si log_mode=true, alors le logarithme naturel de la probabilité est calculée.

error_code

[out] Variable pour stocker le code d'erreur.

result[]

[out] Tableau des valeurs de la fonction de distribution.

© 2000-2024, MetaQuotes Ltd.

MathQuantilePoisson

Pour la *probabilité* spécifiée, la fonction calcule la valeur inverse de la fonction de distribution de Poisson avec le paramètre lambda. En cas d'erreur, retourne NaN.

```
double  MathQuantilePoisson(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  lambda,         // paramètre de la distribution (moyenne)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour la *probabilité* spécifiée, la fonction calcule la valeur inverse de la fonction de distribution de Poisson avec le paramètre lambda. En cas d'erreur, retourne NaN.

```
double  MathQuantilePoisson(
   const double  probability,    // probabilité d'occurrence de la variable aléatoire
   const double  lambda,         // paramètre de la distribution (moyenne)
   int&          error_code      // variable stockant le code d'erreur
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur inverse de la fonction de distribution de Poisson avec le paramètre lambda. En cas d'erreur, retourne false. Analogue à qpois() dans R.

```
double  MathQuantilePoisson(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  lambda,         // paramètre de la distribution (moyenne)
   const bool    tail,           // flag de calcul, si false, alors le calcul est effe
   const bool    log_mode,       // flag de calcul, si log_mode=true, le calcul est ef
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Pour les valeurs de probabilité du tableau *probability[]* spécifié, la fonction calcule la valeur inverse de la fonction de distribution de Poisson avec le paramètre lambda. En cas d'erreur, retourne false.

```
bool  MathQuantilePoisson(
   const double& probability[],  // tableau des valeurs de probabilités de la variable
   const double  lambda,         // paramètre de la distribution (moyenne)
   double&       result[]        // tableau des valeurs des quantiles
   );
```

Paramètres

probability

[in] Probabilité de la variable aléatoire.

probability[]

[in] Tableau des valeurs de probabilités de la variable aléatoire.

lambda

© 2000-2024, MetaQuotes Ltd.

[in] Paramètre de la distribution (moyenne).

tail

[in] Flag de calcul, si tail=false, alors le calcul est effectué pour 1.0-probabilité.

log_mode

[in] Flag de calcul, si log_mode=true, le calcul est effectué pour Exp(probabilité).

error_code

[out] Variable pour récupérer le code d'erreur.

result[]

[out] Tableau de valeurs des quantiles.

© 2000-2024, MetaQuotes Ltd.

MathRandomPoisson

Génère une variable pseudo-alatoire distribuée suivant la loi de distribution de Poisson avec le paramètre lambda. En cas d'erreur, retourne NaN.

```
double  MathRandomPoisson(
   const double  lambda,          // paramètre de la distribution (moyenne)
   int&          error_code       // variable stockant le code d'erreur
   );
```

Génère des variables pseudo-alatoires distribuées suivant la loi de distribution de Poisson avec le paramètre lambda. En cas d'erreur, retourne false. Analogue à rpois() dans R.

```
bool  MathRandomPoisson(
   const double  lambda,          // paramètre de la distribution (moyenne)
   const int     data_count,      // quantité de données nécessaires
   double&       result[]         // tableau des valeurs des variables pseudo-aléatoire
   );
```

Paramètres

lambda

[in] Paramètre de la distribution (moyenne).

error_code

[out] Variable pour stocker le code d'erreur.

data_count

[out] Quantité de données nécessaires.

result[]

[out] Tableau des valeurs des variables pseudo-aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathMomentsPoisson

Calcule les valeurs numériques théoriques des 4 premiers moments de la distribution de Poisson avec le paramètre lambda.

```
double  MathMomentsPoisson(
   const double  lambda,          // paramètre de la distribution (moyenne)
   double&       mean,            // variable pour la moyenne
   double&       variance,        // variable pour la variance
   double&       skewness,        // variable pour le coefficient de dissymétrie (skewr
   double&       kurtosis,        // variable pour le kurtosis
   int&          error_code       // variable pour le code d'erreur
   );
```

Paramètres

lambda

[in] Paramètre de la distribution (moyenne).

mean

[out] Variable pour récupérer la valeur moyenne.

variance

[out] Variable pour récupérer la variance.

skewness

[out] Variable pour récuperer le coefficient de dissymétrie (skewness en anglais).

kurtosis

[out] Variable pour récupérer le kurtosis.

error_code

[out] Variable pour récupérer le code d'erreur.

Valeur de Retour

Retourne true si le calcul des moments a été réalisé avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Sous-fonctions

Groupe de fonctions effectuant des opérations mathématiques de base : calcul des fonctions gamma, beta, factorielle, exponentielle, logarithmes avec différentes bases, racine carrée, etc.

Ils permettent de traiter des valeurs numériques individuelles (réelles et entières) et des tableaux de telles valeurs (avec la sortie des résultats dans un tableau séparé ou dans le tableau d'origine).

Fonction	Description
MathRandomNonZero	Retourne un nombre aléatoire avec une virgule flottante dans l'intervalle de 0.0 à 1.0.
MathMoments	Calcule les quatre premiers moments d'éléments de tableau: moyenne, variance, dissymétrie, kurtosis.
MathPowInt	Elève un nombre à la puissance entière spécifiée.
MathFactorial	Calcule la factorielle de l'entier spécifié.
MathTrunc	Calcule la partie entière du nombre ou du tableau d'éléments spécifiés.
MathRound	Arrondit un nombre ou un tableau de nombres au nombre de décimales spécifié.
MathArctan2	Calcule l'angle dont la tangente est égale au rapport entre les deux nombres spécifiés dans l'intervalle [-pi, pi].
MathGamma	Calcule la valeur de la fonction gamma.
MathGammaLog	Calcule le logarithme de la fonction gamma.
MathBeta	Calcule la valeur de la fonction beta.
MathBetaLog	Calcule le logarithme de la fonction beta.
MathBetaIncomplete	Calcule la valeur de la fonction beta incomplète.
MathGammaIncomplete	Calcule la valeur de la fonction gamma incomplète.
MathBinomialCoefficient	Calcule le coefficient binomial.
MathBinomialCoefficientLog	Calcule le logarithme du coefficient binomial.
MathHypergeometric2F2	Calcule la valeur de la fonction hypergéométrique.
MathSequence	Génère une séquence en fonction des valeurs suivantes : le premier élément, le dernier élément, l'étape de la séquence.

© 2000-2024, MetaQuotes Ltd.

Fonction	Description
MathSequenceByCount	Génère une séquence en fonction des valeurs suivantes : le premier élément, le dernier élément, le nombre d'éléments dans la séquence.
MathReplicate	Génère une séquence répétitive de valeurs.
MathReverse	Génère un tableau de valeurs avec l'ordre inverse des éléments.
MathIdentical	Compare deux tableaux de valeurs et retourne true si tous les éléments correspondent.
MathUnique	Génère un tableau ne contenant que des valeurs uniques.
MathQuickSortAscending	Fonction de tri dans l'ordre ascendant.
MathQuickSortDescending	Fonction de tri dans l'ordre descendant.
MathQuickSort	Fonction de tri.
MathOrder	Génère un tableau avec des permutations selon l'ordre des éléments du tableau après le tri.
MathBitwiseNot	Calcule le résultat de l'opération sur les bits NOT pour les éléments du tableau.
MathBitwiseAnd	Calcule le résultat de l'opération sur les bits AND pour les éléments des tableaux.
MathBitwiseOr	Calcule le résultat de l'opération sur les bits OR pour les éléments des tableaux.
MathBitwiseXor	Calcule le résultat de l'opération sur les bits XOR pour les éléments des tableaux.
MathBitwiseShiftL	Calcule le résultat de l'opération sur les bits SHL pour les éléments du tableau.
MathBitwiseShiftR	Calcule le résultat de l'opération sur les bits SHR pour les éléments du tableau.
MathCumulativeSum	Génère un tableau avec les sommes cumulées.
MathCumulativeProduct	Génère un tableau avec les produits cumulés.
MathCumulativeMin	Génère un tableau avec les minima cumulatifs.
MathCumulativeMax	Génère un tableau avec les maxima cumulatifs.
MathSin	Calcule les valeurs de la fonction sin(x) pour les éléments du tableau.

© 2000-2024, MetaQuotes Ltd.

Fonction	Description
MathCos	Calcule les valeurs de la fonction cos(x) pour les éléments du tableau.
MathTan	Calcule les valeurs de la fonction tan(x) pour les éléments du tableau.
MathArcsin	Calcule les valeurs de la fonction arcsin(x) pour les éléments du tableau.
MathArccos	Calcule les valeurs de la fonction arccos(x) pour les éléments du tableau.
MathArctan	Calcule les valeurs de la fonction arctan(x) pour les éléments du tableau.
MathSinPi	Calcule les valeurs de la fonction sin(pi*x) pour les éléments du tableau.
MathCosPi	Calcule les valeurs de la fonction cos(pi*x) pour les éléments du tableau.
MathTanPi	Calcule les valeurs de la fonction tan(pi*x) pour les éléments du tableau.
MathAbs	Calcule les valeurs absolues des éléments du tableau.
MathCeil	Renvoie les entiers plus grands les plus proches pour les éléments du tableau.
MathFloor	Renvoie les entiers plus petits les plus proches pour les éléments du tableau.
MathSqrt	Calcule les racines carrées des éléments du tableau.
MathExp	Calcule les valeurs de la fonction exp(x) pour les éléments du tableau.
MathPow	Calcule les valeurs de la fonction pow(x, power) pour les éléments du tableau.
MathLog	Calcule les valeurs de la fonction log(x) pour les éléments du tableau.
MathLog2	Calcule le logarithme à base 2 des éléments du tableau.
MathLog10	Calcule le logarithme à base 10 des éléments du tableau.
MathDifference	Génère un tableau avec les différences d'éléments de y[i]=x[i+lag]-x[i].
MathSample	Génère un échantillon aléatoire à partir des éléments du tableau.

© 2000-2024, MetaQuotes Ltd.

Fonction	Description
MathTukeySummary	Calcule le résumé des cinq nombres de Tukey pour les éléments du tableau.
MathRange	Calcule le minima et le maxima des éléments du tableau.
MathMin	Retourne la valeur minimum de tous les éléments du tableau.
MathMax	Retourne la valeur maximum de tous les éléments du tableau.
MathSum	Retourne la somme de tous les éléments du tableau.
MathProduct	Retourne le produit de tous les éléments du tableau.
MathStandardDeviation	Calcule l'écart-type des éléments du tableau.
MathAverageDeviation	Calcule l'écart absolu moyen des éléments du tableau.
MathMedian	Calcule la valeur médiane des éléments du tableau.
MathMean	Calcule les valeurs moyennes des éléments du tableau.
MathVariance	Calcule la variance des éléments du tableau.
MathSkewness	Calcule l'asymétrie des éléments du tableau.
MathKurtosis	Calcule la kurtosis des éléments du tableau.
MathLog1p	Calcule les valeurs de la fonction log(1+x) pour les éléments du tableau.
MathExpm1	Calcule les valeurs de la fonction exp(x)-1 pour les éléments du tableau.
MathSinh	Calcule les valeurs de la fonction sinh(x) pour les éléments du tableau.
MathCosh	Calcule les valeurs de la fonction cosh(x) pour les éléments du tableau.
MathTanh	Calcule les valeurs de la fonction tanh(x) pour les éléments du tableau.
MathArcsinh	Calcule les valeurs de la fonction arcsinh(x) pour les éléments du tableau.
MathArccosh	Calcule les valeurs de la fonction arccosh(x) pour les éléments du tableau.

© 2000-2024, MetaQuotes Ltd.

Fonction	Description
MathArctanh	Calcule les valeurs de la fonction arctanh(x) pour les éléments du tableau.
MathSignif	Arrondit une valeur au nombre spécifié de chiffres dans la mantisse.
MathRank	Calcule les rangs des éléments de tableau.
MathCorrelationPearson	Calcule le coefficient de corrélation de Pearson.
MathCorrelationSpearman	Calcule le coefficient de corrélation de Spearman.
MathCorrelationKendall	Calcule le coefficient de corrélation de Kendall.
MathQuantile	Calcule des quantiles d'échantillons correspondant aux probabilités spécifiées.
MathProbabilityDensityEmpirical	Calcule la fonction de densité de probabilité empirique pour les valeurs aléatoires.
MathCumulativeDistributionEmpirical	Calcule la fonction de répartition cumulative empirique pour les valeurs aléatoires.

© 2000-2024, MetaQuotes Ltd.

MathRandomNonZero

Retourne un nombre aléatoire avec une virgule flottante dans l'intervalle de 0.0 à 1.0.

```
double  MathRandomNonZero()
```

Valeur de Retour

Un nombre aléatoire avec une virgule flottante dans l'intervalle de 0.0 à 1.0.

© 2000-2024, MetaQuotes Ltd.

MathMoments

Calcule les quatre premiers moments d'éléments de tableau: moyenne, variance, dissymétrie, kurtosis.

```
bool  MathMoments(
   const double&  array[],            // tableau des valeurs
   double&        mean,               // variable pour la moyenne
   double&        variance,           // variable pour la variance
   double&        skewness,           // variable pour le coefficient de dissymétrie
   double&        kurtosis,           // variable pour le kurtosis
   const int      start=0,            // indice initial
   const int      count=WHOLE_ARRAY   // le nombre d'éléments
  )
```

Paramètres

array[]

[in] Tableau de valeurs.

mean

[out] Variable pour la moyenne (1er moment).

variance

[out] Variable pour la variance (2ème moment).

skewness

[out] Variable pour le coefficient de dissymétrie (3ème moment).

kurtosis

[out] Variable pour le kurtosis (4ème moment).

start=0

[in] Indice de départ pour le calcul.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour le calcul.

Valeur de Retour

Retourne true si les moments ont été calculés avec succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathPowInt

Elève un nombre à la puissance entière spécifiée.

```
double  MathPowInt(
   const double  x,       // valeur du nombre
   const int     power    // puissance à laquelle élever
  )
```

Paramètres

x

[in] Nombre à virgule flottante à double précision devant être élevé à la puissance.

power

[in] Entier spécifiant la puissance.

Valeur de Retour

Le nombre *x* élevé à la puissance spécifiée.

© 2000-2024, MetaQuotes Ltd.

MathFactorial

Calcule la factorielle de l'entier spécifié.

```
double  MathFactorial(
   const int  n   // valeur du nombre
  )
```

Paramètres

n

[in] Nombre entier pour lequel la factorielle est calculée.

Valeur de Retour

La factorielle du numéro.

© 2000-2024, MetaQuotes Ltd.

MathTrunc

Calcule la partie entière du nombre ou du tableau d'éléments spécifiés.

Version pour travailler avec un nombre à virgule flottante à double précision :

```
double  MathTrunc(
   const double   x          // valeur du nombre
  )
```

Valeur de Retour

La partie entière du nombre spécifié.

Version pour travailler avec un tableau de nombres à virgule flottante à double précision. Les résultats sont retournés dans un nouveau tableau :

```
bool  MathTrunc(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

Version pour travailler avec un tableau de nombres à virgule flottante à double précision. Les résultats sont retournés dans le tableau d'origine :

```
bool  MathTrunc(
   double&        array[]    // tableau des valeurs
  )
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

Paramètres

x

[in] Nombre à virgule flottante à double précision dont la partie entière est demandée.

array[]

[in] Tableau de nombres à virgule flottante à double précision dont les parties entières sont demandées.

array[]

[out] Tableau des valeurs de sortie.

result[]

[out] Tableau des valeurs de sortie.

© 2000-2024, MetaQuotes Ltd.

MathRound

Arrondit un nombre à virgule flottante à double précision ou un tableau de tels nombres au nombre de décimales spécifié.

Version pour arrondir un nombre à virgule flottante à double précision au nombre de décimales spécifié :

```
double  MathRound(
   const double   x,          // valeur du nombre
   const int      digits      // le nombre de chiffres décimaux
  )
```

Valeur de Retour

Le nombre le plus proche du paramètre x, avec le nombre de décimales égal au paramètre digits.

Version pour arrondir un tableau de nombres à virgule flottante à double précision au nombre de décimales spécifié. Les résultats sont retournés dans un nouveau tableau.

```
bool  MathRound(
   const double&  array[],    // tableau des valeurs
   int            digits,     // le nombre de chiffres décimaux
   double&        result[]    // tableau des résultats
  )
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

Version pour arrondir un tableau de nombres à virgule flottante à double précision au nombre de décimales spécifié. Les résultats sont retournés dans le tableau d'origine.

```
bool  MathRound(
   double&        array[],    // tableau des valeurs
   int            digits      // le nombre de chiffres décimaux
  )
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

Paramètres

x

[in] Nombre à virgule flottante à double précision devant être arrondi.

digits

[in] Le nombre de décimales dans la valeur de retour.

array[]

[in] Tableau de nombres à virgule flottante à double précision devant être arrondis.

array[]

© 2000-2024, MetaQuotes Ltd.

[out] Tableau des valeurs de sortie.

result[]

[out] Tableau des valeurs de sortie.

© 2000-2024, MetaQuotes Ltd.

MathArctan2

Renvoie l'arctangente du quotient de deux arguments (x, y).

Version pour travailler avec le rapport des deux nombres spécifiés (x, y) :

```
double  MathArctan2(
   const double       y,          // Coordonnée Y
   const double       x           // Coordonnée X
  )
```

Valeur de Retour

Angle θ, mesuré en radians, de sorte que -π≤θ≤π et tan (θ) = y ou x, où (x, y) est un point dans un système de coordonnées cartésiennes.

Version pour travailler avec le rapport des couples d'éléments des matrices x et y :

```
bool  MathArctan2(
   const double&      x[],        // tableau des valeurs x
   const double&      y[],        // tableau des valeurs y
   double&            result[]    // tableau des résultats
  )
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

Paramètres

y

[in] La coordonnée Y du point.

x

[in] La coordonnée X du point.

x[]

[in] Tableau des coordonnées X des points.

y[]

[in] Tableau des coordonnées Y des points.

result[]

[out] Tableau pour retourner les résultats

Notes

Veuillez noter que :

- Pour (x, y) dans le quadrant 1, la valeur de retour sera : 0 < θ < π/2.
- Pour (x, y) dans le quadrant 2, la valeur de retour sera : π/2 < θ ≤ π.
- Pour (x, y) dans le quadrant 3, la valeur de retour sera : -π < θ < -π/2.
- Pour (x, y) dans le quadrant 4, la valeur de retour sera : -π/2 < θ < 0.

© 2000-2024, MetaQuotes Ltd.

La valeur de retour pour les points situés en dehors de ces quadrants est située ci-dessous.

- Si y est égal à 0 et que x n'est pas négatif, alors θ = 0.
- Si y est égal à 0 et que x est négatif, alors θ = π.
- Si y est positif et que x est égal à 0, alors θ = π/2.
- Si y est négatif et que x est égal à 0, alors θ = -π/2.
- Si y est égal à 0 et que x est égal à 0, alors θ = -π/2.

Si la valeur du paramètre x ou y est NaN, ou si les valeurs des paramètres x et y sont égales à la valeur PositiveInfinity ou NegativeInfinity, la méthode renvoie la valeur NaN.

© 2000-2024, MetaQuotes Ltd.

MathGamma

Calcule la valeur de la fonction gamma pour l'argument réel *x*.

```
double  MathGamma(
   const double  x       // argument de la fonction
  )
```

Paramètres

x

[in] L'argument réel de la fonction.

Valeur de Retour

La valeur de la fonction gamma.

© 2000-2024, MetaQuotes Ltd.

MathGammaLog

Calcule le logarithme de la fonction gamma pour l'argument réel x.

```
double  MathGammaLog(
   const double  x      // argument de la fonction
  )
```

Paramètres

x

[in] L'argument réel de la fonction.

Valeur de Retour

Logarithme de la fonction.

© 2000-2024, MetaQuotes Ltd.

MathBeta

Calcule la valeur de la fonction beta pour les arguments réels a et b.

```
double  MathBeta(
   const double  a,       // le premier argument de la fonction
   const double  b        // le deuxième argument de la fonction
  )
```

Paramètres

a

[in] L'argument a de la fonction.

b

[in] L'argument b de la fonction.

Valeur de Retour

Valeur de la fonction.

© 2000-2024, MetaQuotes Ltd.

MathBetaLog

Calcule le logarithme de la fonction beta pour les arguments réels a et b.

```
double  MathBetaLog(
   const double  a,       // le premier argument de la fonction
   const double  b        // le deuxième argument de la fonction
  )
```

Paramètres

a

[in] L'argument a de la fonction.

b

[in] L'argument b de la fonction.

Valeur de Retour

Logarithme de la fonction.

© 2000-2024, MetaQuotes Ltd.

MathBetaIncomplete

Calcule la valeur de la fonction beta incomplète.

```
double  MathBetaIncomplete(
   const double  x,       // argument de la fonction
   const double  p,       // le premier paramètre de la fonction
   const double  q        // le deuxième paramètre de la fonction
  )
```

Paramètres

x

[in] L'argument de la fonction.

p

[in] Le premier paramètre de la fonction beta, doit être >0.0.

q

[in] Le deuxième paramètre de la fonction beta, doit être >0.0.

Valeur de Retour

Valeur de la fonction.

© 2000-2024, MetaQuotes Ltd.

MathGammaIncomplete

Calcule la valeur de la fonction gamma incomplète.

```
double  MathGammaIncomplete(
   double  x,          // argument de la fonction
   double  alpha       // paramètre de la fonction
  )
```

Paramètres

x

[in] L'argument de la fonction.

alpha

[in] Le paramètre de la fonction gamma incomplète.

Valeur de Retour

Valeur de la fonction.

© 2000-2024, MetaQuotes Ltd.

MathBinomialCoefficient

Calcule le coefficient binomial : C(n,k)=n!/(k!*(n-k)!).

```
long  MathBinomialCoefficient(
   const int  n,       // le nombre total d'éléments
   const int  k        // le nombre combinatoire d'éléments
  )
```

Paramètres

n

[in] Le nombre d'éléments.

k

[in] Le nombre d'éléments pour chaque combinaison.

Valeur de Retour

Le nombre de combinaisons de K parmi N.

© 2000-2024, MetaQuotes Ltd.

MathBinomialCoefficientLog

Calcule le logarithme du coefficient binomial : Log(C(n,k))=Log(n!/(k!*(n-k)!)).

Version avec des arguments de type integer :

```
double  MathBinomialCoefficientLog(
   const int     n,       // le nombre total d'éléments
   const int     k        // le nombre combinatoire d'éléments
  )
```

Version avec des arguments réels :

```
double  MathBinomialCoefficientLog(
   const double  n,       // le nombre total d'éléments
   const double  k        // le nombre d'éléments en combinaison
  )
```

Paramètres

n

[in] Le nombre d'éléments.

k

[in] Le nombre d'éléments pour chaque combinaison.

Valeur de Retour

Le logarithme de C(n,k).

© 2000-2024, MetaQuotes Ltd.

MathHypergeometric2F2

Calcule la valeur de la fonction Hypergeometric_2F2 (a, b, c, d, z) selon la méthode de Taylor.

```
double  MathHypergeometric2F2(
   const double  a,      // le premier paramètre de la fonction
   const double  b,      // le deuxième paramètre de la fonction
   const double  c,      // le troisième paramètre de la fonction
   const double  d,      // le quatrième paramètre de la fonction
   const double  z       // le cinquième paramètre de la fonction
  )
```

Paramètres

a

[in] Le premier paramètre de la fonction.

b

[in] Le deuxième paramètre de la fonction.

c

[in] Le troisième paramètre de la fonction.

d

[in] Le quatrième paramètre de la fonction.

z

[in] Le cinquième paramètre de la fonction.

Valeur de Retour

Valeur de la fonction.

© 2000-2024, MetaQuotes Ltd.

MathSequence

Génère une séquence de valeurs en fonction des valeurs suivantes : le premier élément, le dernier élément, l'étape de la séquence.

Version pour travailler avec des valeurs réelles :

```
bool  MathSequence(
   const double  from,        // valeur de départ
   const double  to,          // valeur finale
   const double  step,        // étape
   double&       result[]     // tableau des résultats
  )
```

Version pour travailler avec des valeurs entières :

```
bool  MathSequence(
   const int     from,        // valeur de départ
   const int     to,          // valeur finale
   const int     step,        // étape
   int&          result[]     // tableau des résultats
  )
```

Paramètres

from

[in] La première valeur de la séquence

to

[in] La dernière valeur de la séquence

step

[in] L'étape de la séquence.

result[]

[out] Tableau pour retourner la séquence.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathSequenceByCount

Génère une séquence de valeurs en fonction des valeurs suivantes : le premier élément, le dernier élément, le nombre d'éléments dans la séquence.

Version pour travailler avec des valeurs réelles :

```
bool  MathSequenceByCount(
   const double  from,        // valeur de départ
   const double  to,          // valeur finale
   const int     count,       // compte
   double&       result[]     // tableau des résultats
  )
```

Version pour travailler avec des valeurs entières :

```
bool  MathSequenceByCount(
   const int     from,        // valeur de départ
   const int     to,          // valeur finale
   const int     count,       // compte
   int&          result[]     // tableau des résultats
  )
```

Paramètres

from

[in] La première valeur de la séquence.

to

[in] La dernière valeur de la séquence.

count

[in] Le nombre d'éléments dans la séquence.

result[]

[out] Tableau pour retourner la séquence.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathReplicate

Génère une séquence répétitive de valeurs.

Version pour travailler avec des valeurs réelles :

```
bool  MathReplicate(
   const double&  array[],    // tableau des valeurs
   const int      count,      // nombre de répétitions
   double&        result[]    // tableau des résultats
  )
```

Version pour travailler avec des valeurs entières :

```
bool  MathReplicate(
   const int&     array[],    // tableau des valeurs
   const int      count,      // nombre de répétitions
   int&           result[]    // tableau des résultats
  )
```

Paramètres

array[]

[in] Tableau pour générer une séquence.

count

[in] Le nombre de répétitions du tableau dans la séquence.

result[]

[out] Tableau pour retourner la séquence.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathReverse

Génère un tableau de valeurs avec l'ordre inverse des éléments.

Version pour travailler avec des valeurs réelles retournant les résultats dans un nouveau tableau :

```
bool  MathReverse(
   const double&  array[],      // tableau des valeurs
   double&        result[]      // tableau des résultats
  )
```

Version pour travailler avec des valeurs entières retournant les résultats dans un nouveau tableau :

```
bool  MathReverse(
   const int&     array[],      // tableau des valeurs
   int&           result[]      // tableau des résultats
  )
```

Version pour travailler avec des valeurs réelles retournant les résultats dans le tableau d'origine :

```
bool  MathReverse(
   double&        array[]       // tableau des valeurs
  )
```

Version pour travailler avec des valeurs entières retournant les résultats dans le tableau d'origine :

```
bool  MathReverse(
   int&           array[]       // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

array[]

[out] Tableau de sortie avec l'ordre inverse des valeurs.

result[]

[out] Tableau de sortie avec l'ordre inverse des valeurs.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathIdentical

Compare deux tableaux de valeurs et retourne true si tous les éléments correspondent.

Version pour travailler avec des tableaux de valeurs réelles :

```
bool  MathIdentical(
   const double&  array1[],      // le premier tableau de valeurs
   const double&  array2[]       // le deuxième tableau de valeurs
  )
```

Version pour travailler avec des tableaux de valeurs entières :

```
bool  MathIdentical(
   const int&     array1[],      // le premier tableau de valeurs
   const int&     array2[]       // le deuxième tableau de valeurs
  )
```

Paramètres

array1[]

[in] Le premier tableau de valeurs.

array2[]

[in] Le deuxième tableau à comparer.

Valeur de Retour

Retourne true si les tableaux sont égaux, false sinon.

© 2000-2024, MetaQuotes Ltd.

MathUnique

Génère un tableau ne contenant que des valeurs uniques.

Version pour travailler avec des valeurs réelles :

```
bool  MathUnique(
   const double&   array[],   // tableau des valeurs
   double&         result[]   // tableau des résultats
  )
```

Version pour travailler avec des valeurs entières :

```
bool  MathUnique(
   const int&     array[],      // tableau des valeurs
   int&           result[]      // tableau des résultats
  )
```

Paramètres

array[]

[in] Le tableau source.

result[]

[out] Tableau pour retourner les valeurs uniques.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathQuickSortAscending

La fonction pour le tri ascendant simultané des tableaux array[] et indices[] à l'aide de l'algorithme QuickSort.

```
void  MathQuickSortAscending(
   double&  array[],      // tableau des valeurs
   int&     indices[],    // tableau d'indices
   int      first,        // valeur de départ
   int      last          // valeur finale
  )
```

Paramètres

array[]

[in][out] Tableau à trier.

indices[]

[in][out] Tableau pour stocker les indices du tableau d'origine.

first

[in] Indice de l'élément de départ du tri.

last

[in] Indice de l'élément de fin du tri.

© 2000-2024, MetaQuotes Ltd.

MathQuickSortDescending

La fonction pour le tri descendant simultané des tableaux array[] et indices[] à l'aide de l'algorithme QuickSort.

```
void  MathQuickSortDescending(
   double&  array[],      // tableau des valeurs
   int&     indices[],    // tableau d'indices
   int      first,        // valeur de départ
   int      last          // valeur finale
  )
```

Paramètres

array[]

[in][out] Tableau à trier.

indices[]

[in][out] Tableau pour stocker les indices du tableau d'origine.

first

[in] Indice de l'élément de départ du tri.

last

[in] Indice de l'élément de fin du tri.

© 2000-2024, MetaQuotes Ltd.

MathQuickSort

La fonction pour le tri simultané des tableaux array[] et indices[] à l'aide de l'algorithme QuickSort.

```
void  MathQuickSort(
   double&  array[],      // tableau des valeurs
   int&     indices[],    // tableau d'indices
   int      first,        // valeur de départ
   int      last,         // valeur finale
   int      mode          // direction
  )
```

Paramètres

array[]

[in][out] Tableau à trier.

indices[]

[in][out] Tableau pour stocker les indices du tableau d'origine.

first

[in] Indice de l'élément de départ du tri.

last

[in] Indice de l'élément de fin du tri.

mode

[in] Direction du tri (>0 ascendant; sinon, descendant).

© 2000-2024, MetaQuotes Ltd.

MathOrder

Génère un tableau d'entiers avec des permutations selon l'ordre des éléments du tableau après le tri.

Version pour travailler avec un tableau de valeurs réelles :

```
bool  MathOrder(
   const double&  array[],   // tableau des valeurs
   int&           result[]   // tableau des résultats
  )
```

Version pour travailler avec un tableau de valeurs entières :

```
bool  MathOrder(
   const int&     array[],   // tableau des valeurs
   int&           result[]   // tableau des résultats
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau pour retourner les indices triés.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseNot

Calcule le résultat de l'opération sur les bits NOT pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathBitwiseNot(
   const int&  array[],   // tableau des valeurs
   int&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathBitwiseNot(
   int&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

array[]

[out] Tableau des valeurs de sortie.

result[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseAnd

Calcule le résultat de l'opération sur les bits AND pour les tableaux spécifiés.

```
bool  MathBitwiseAnd(
   const int&  array1[],  // le premier tableau de valeurs
   const int&  array2[],  // le deuxième tableau de valeurs
   int&        result[]   // tableau des résultats
  )
```

Paramètres

array1[]

[in] Le premier tableau de valeurs.

array2[]

[in] Le deuxième tableau de valeurs.

result[]

[out] Tableau pour retourner les résultats.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseOr

Calcule le résultat de l'opération sur les bits OR pour les tableaux spécifiés.

```
bool  MathBitwiseOr(
   const int&  array1[],  // le premier tableau de valeurs
   const int&  array2[],  // le deuxième tableau de valeurs
   int&        result[]   // tableau des résultats
  )
```

Paramètres

array1[]

[in] Le premier tableau de valeurs.

array2[]

[in] Le deuxième tableau de valeurs.

result[]

[out] Tableau pour retourner les résultats.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseXor

Calcule le résultat de l'opération sur les bits XOR pour les tableaux spécifiés.

```
bool  MathBitwiseXor(
   const int&  array1[],  // le premier tableau de valeurs
   const int&  array2[],  // le deuxième tableau de valeurs
   int&        result[]   // tableau des résultats
  )
```

Paramètres

array1[]

[in] Le premier tableau de valeurs.

array2[]

[in] Le deuxième tableau de valeurs.

result[]

[out] Tableau pour retourner les résultats.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseShiftL

Calcule le résultat de l'opération sur les bits SHL (bitwise shift left - décalage vers la gauche des bits) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathBitwiseShiftL(
   const int&  array[],     // tableau des valeurs
   const int   n,           // valeur de décalage
   int&        result[]     // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathBitwiseShiftL(
   int&        array[],     // tableau des valeurs
   const int   n            // valeur de décalage
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

n

[in] Le nombre de décalage de bits.

array[]

[out] Tableau des valeurs de sortie.

result[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseShiftR

Calcule le résultat de l'opération sur les bits SHR (bitwise shift right - décalage vers la droite des bits) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathBitwiseShiftR(
   const int&  array[],     // tableau des valeurs
   const int   n,           // valeur de décalage
   int&        result[]     // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathBitwiseShiftR(
   int&        array[],     // tableau des valeurs
   const int   n            // valeur de décalage
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

n

[in] Le nombre de décalage de bits.

array[]

[out] Tableau des valeurs de sortie.

result[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeSum

Génère un tableau avec les sommes cumulées.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathCumulativeSum(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathCumulativeSum(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

array[]

[out] Tableau des valeurs de sortie.

result[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeProduct

Génère un tableau avec les produits cumulés.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathCumulativeProduct(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathCumulativeProduct(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeMin

Génère un tableau avec les minima cumulatifs.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathCumulativeMin(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathCumulativeMin(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeMax

Génère un tableau avec les maxima cumulatifs.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathCumulativeMax(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathCumulativeMax(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathSin

Calcule les valeurs de la fonction sin(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathSin(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathSin(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCos

Calcule les valeurs de la fonction cos(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathCos(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathCos(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathTan

Calcule les valeurs de la fonction tan(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathTan(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathTan(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathArcsin

Calcule les valeurs de la fonction arcsin(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathArcsin(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathArcsin(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathArccos

Calcule les valeurs de la fonction arccos(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathArccos(
   const double&   array[],     // tableau des valeurs
   double&         result[]     // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathArccos(
   double&         array[]     // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathArctan

Calcule les valeurs de la fonction arctan(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathArctan(
   const double&   array[],   // tableau des valeurs
   double&         result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathArctan(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathSinPi

Calcule les valeurs de la fonction sin(pi*x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathSinPi(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathSinPi(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCosPi

Calcule les valeurs de la fonction cos(pi*x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathCosPi(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathCosPi(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathTanPi

Calcule les valeurs de la fonction tan(pi*x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathTanPi(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathTanPi(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathAbs

Calcule les valeurs absolues des éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathAbs(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathAbs(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCeil

Renvoie les entiers plus grands les plus proches pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathCeil(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathCeil(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathFloor

Renvoie les entiers plus petits les plus proches pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathFloor(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathFloor(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathSqrt

Calcule les racines carrées des éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathSqrt(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathSqrt(
   double&         array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathExp

Calcule les valeurs de la fonction exp(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathExp(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathExp(
   double&       array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathPow

Calcule les valeurs de la fonction pow(x, power) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathPow(
   const double&  array[],    // tableau des valeurs
   const double   power,      // puissance
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathPow(
   double&        array[],    // tableau des valeurs
   const double   power       // puissance
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathLog

Calcule les valeurs de la fonction log(x) pour les éléments du tableau.

Version calculant le logarithme naturel avec sortie des résultats dans un nouveau tableau.

```
bool  MathLog(
   const double&  array[],      // tableau des valeurs
   double&        result[]      // tableau des résultats
  )
```

Version calculant le logarithme naturel avec sortie des résultats dans le tableau d'origine.

```
bool  MathLog(
   double&        array[]       // tableau des valeurs
  )
```

Version calculant le logarithme dans une base spécifiée avec sortie des résultats dans un nouveau tableau.

```
bool  MathLog(
   const double&   array[],     // tableau des valeurs
   const double    base,        // base du logarithme
   double&         result[]     // tableau des résultats
  )
```

Version calculant le logarithme dans une base spécifiée avec sortie des résultats dans le tableau d'origine.

```
bool  MathLog(
   double&         array[],     // tableau des valeurs
   const double    base         // base du logarithme
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

base

[in] La base du logarithme.

array[]

[out] Tableau des valeurs de sortie.

result[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathLog2

Calcule le logarithme à base 2 des éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathLog2(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathLog2(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathLog10

Calcule le logarithme à base 10 des éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathLog10(
   const double&   array[],   // tableau des valeurs
   double&         result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathLog10(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathLog1p

Calcule les valeurs de la fonction log(1+x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathLog1p(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathLog1p(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathDifference

Génère un tableau avec les différences d'éléments de y[i]=x[i+lag]-x[i].

Version pour une seule génération d'un tableau de valeurs réelles :

```
bool  MathDifference(
   const double  &array[],      // tableau des valeurs
   const int     lag,           // décalage
   double        &result[]      // tableau des résultats
  )
```

Version pour une seule génération d'un tableau de valeurs entières :

```
bool  MathDifference(
   const int     &array[],      // tableau des valeurs
   const int     lag,           // décalage
   int           &result[]      // tableau des résultats
  )
```

Version pour la génération itérée d'un tableau de valeurs réelles (le nombre d'itérations est défini dans les paramètres d'entrée) :

```
bool  MathDifference(
   const double  &array[],      // tableau des valeurs
   const int     lag,           // décalage
   const int     differences,   // nombre d'itérations
   double        &result[]      // tableau des résultats
  )
```

Version pour la génération itérée d'un tableau de valeurs entières (le nombre d'itérations est défini dans les paramètres d'entrée) :

```
bool  MathDifference(
   const int&    array[],       // tableau des valeurs
   const int     lag,           // décalage
   const int     differences,   // nombre d'itérations
   int&          result[]       // tableau des résultats
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

lag

[in] Paramètre de décalage.

differences

[in] Le nombre d'itérations.

result[]

© 2000-2024, MetaQuotes Ltd.

[out] Tableau pour retourner les résultats.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathSample

Génère un échantillon aléatoire à partir des éléments du tableau.

Version pour travailler avec un tableau de valeurs réelles :

```
bool  MathSample(
   const double&  array[],          // tableau des valeurs
   const int      count,            // compte
   double&        result[]          // tableau des résultats
  )
```

Version pour travailler avec un tableau de valeurs entières :

```
bool  MathSample(
   const int&     array[],          // tableau des valeurs
   const int      count,            // compte
   int&           result[]          // tableau des résultats
  )
```

Version pour travailler avec un tableau de valeurs réelles. Il est possible d'obtenir un échantillon avec remplacement :

```
bool  MathSample(
   const double&  array[],          // tableau des valeurs
   const int      count,            // compte
   const bool     replace,          // flag
   double&        result[],         // tableau des résultats
  )
```

Version pour travailler avec un tableau de valeurs entières. Il est possible d'obtenir un échantillon avec remplacement :

```
bool  MathSample(
   const int&     array[],          // tableau des valeurs
   const int      count,            // compte
   const bool     replace,          // flag
   int&           result[]          // tableau des résultats
  )
```

Version pour travailler avec un tableau de valeurs réelles, pour lesquelles les probabilités d'échantillonnage sont définies.

```
bool  MathSample(
   const double&  array[],          // tableau des valeurs
   double&        probabilities[],  // tableau des probabilités
   const int      count,            // compte
   double&        result[]          // tableau des résultats
  )
```

Version pour travailler avec un tableau de valeurs entières, pour lesquelles les probabilités d'échantillonnage sont définies.

© 2000-2024, MetaQuotes Ltd.

```
bool  MathSample(
   const int&     array[],           // tableau des valeurs
   double&        probabilities[],   // tableau des probabilités<
   const int      count,             // compte
   int&           result[]           // tableau des résultats
  )
```

Version pour travailler avec un tableau de valeurs réelles, pour lesquelles les probabilités d'échantillonnage sont définies. Il est possible d'obtenir un échantillon avec remplacement :

```
bool  MathSample(
   const double&  array[],           // tableau des valeurs
   double&        probabilities[],   // tableau des probabilités
   const int      count,             // compte
   const bool     replace,           // flag
   double&        result[]           // tableau des résultats
  )
```

Version pour travailler avec un tableau de valeurs entières, pour lesquelles les probabilités d'échantillonnage sont définies. Il est possible d'obtenir un échantillon avec remplacement :

```
bool  MathSample(
   const int&     array[],           // tableau des valeurs
   double&        probabilities[],   // tableau de probabilités
   const int      count,             // compte
   const bool     replace,           // flag
   int&           result[]           // tableau des résultats
  )
```

Paramètres

array[]

[in] Tableau de valeurs entières.

probabilities[]

[in] Tableau de probabilités pour l'échantillonnage des éléments.

count

[in] Le nombre d'éléments.

replace

[in] Paramètre permettant d'échantillonner avec remplacement.

result[]

[out] Tableau pour retourner les résultats.

Valeur de Retour

Retourne true en cas de succès, sinon false.

Note

© 2000-2024, MetaQuotes Ltd.

L'argument replace=true permet d'effectuer un échantillonnage aléatoire des éléments avec remplacement avec la séquence d'origine.

© 2000-2024, MetaQuotes Ltd.

MathTukeySummary

Calcule le résumé des cinq nombres de Tukey (minimum, quartile inférieur, médiane, quartile supérieur, maximum) pour les éléments du tableau.

```
bool  MathTukeySummary(
   const double&  array[],       // tableau des valeurs
   const bool     removeNAN,     // flag
   double&        minimum,       // valeur minimum
   double&        lower_hinge,   // quartile inférieur
   double&        median,        // valeur médiane
   double&        upper_hinge,   // quartile supérieur
   double&        maximum        // valeur maximum
  )
```

Paramètres

array[]

[in] Tableau de valeurs réelles.

removeNAN

[in] Flag indiquant si les valeurs non numériques doivent être enlevées.

minimum

[out] Variable pour stocker la valeur minimum.

lower_hinge

[out] Variable pour stocker le quartile inférieur.

median

[out] Variable pour stocker la valeur médiane.

upper_hinge

[out] Variable pour stocker le quartile supérieur.

maximum

[out] Variable pour stocker la valeur maximum.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathRange

Calcule le minima et le maxima des éléments du tableau.

```
bool  MathRange(
   const double&  array[],  // tableau des valeurs
   double&        min,      // valeur minimum
   double&        max       // valeur maximum
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

min

[out] Variable pour stocker la valeur minimum.

max

[out] Variable pour stocker la valeur maximum.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathMin

Retourne la valeur minimum de tous les éléments du tableau.

```
double  MathMin(
   const double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

La valeur minimum.

© 2000-2024, MetaQuotes Ltd.

MathMax

Retourne la valeur maximum de tous les éléments du tableau.

```
double  MathMax(
   const double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

La valeur maximum.

© 2000-2024, MetaQuotes Ltd.

MathSum

Retourne la somme de tous les éléments du tableau.

```
double  MathSum(
   const double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

La somme des éléments.

© 2000-2024, MetaQuotes Ltd.

MathProduct

Retourne le produit de tous les éléments du tableau.

```
double  MathProduct(
   const double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

Le produit des éléments.

© 2000-2024, MetaQuotes Ltd.

MathStandardDeviation

Calcule l'écart-type des éléments du tableau.

```
double  MathStandardDeviation(
   const double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

Ecart-type.

© 2000-2024, MetaQuotes Ltd.

MathAverageDeviation

La fonction calcule l'écart absolu moyen des éléments du tableau.

```
double  MathAverageDeviation(
   const double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

L'écart absolu moyen des éléments du tableau.

© 2000-2024, MetaQuotes Ltd.

MathMedian

Calcule la valeur médiane des éléments du tableau.

```
double  MathMedian(
   double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

La valeur médiane.

© 2000-2024, MetaQuotes Ltd.

MathMean

Calcule les valeurs moyennes des éléments du tableau.

```
double  MathMean(
   const double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

La valeur moyenne.

© 2000-2024, MetaQuotes Ltd.

MathVariance

La fonction calcule la variance (deuxième moment) des éléments du tableau.

```
double  MathVariance(
   const double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

La valeur de la variance.

© 2000-2024, MetaQuotes Ltd.

MathSkewness

La fonction calcule le coefficient de dissymétrie (troisième moment) des éléments du tableau.

```
double  MathSkewness(
   const double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

Le coefficient de dissymétrie.

© 2000-2024, MetaQuotes Ltd.

MathKurtosis

La fonction calcule le kurtosis (quatrième moment) des éléments du tableau.

```
double  MathKurtosis(
   const double&  array[]   // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

Valeur de Retour

Kurtosis.

© 2000-2024, MetaQuotes Ltd.

MathExpm1

Calcule les valeurs de la fonction exp(x)-1 pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathExpm1(
   const double&   array[],   // tableau des valeurs
   double&         result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathExpm1(
   double&         array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathSinh

Calcule les valeurs de la fonction sinh(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathSinh(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathSinh(
   double&       array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCosh

Calcule les valeurs de la fonction cosh(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathCosh(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathCosh(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathTanh

Calcule les valeurs de la fonction tanh(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathTanh(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathTanh(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathArcsinh

Calcule les valeurs de la fonction arcsinh(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathArcsinh(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathArcsinh(
   double&         array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathArccosh

Calcule les valeurs de la fonction arccosh(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathArccosh(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathArccosh(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathArctanh

Calcule les valeurs de la fonction arctanh(x) pour les éléments du tableau.

Version retournant les résultats dans un nouveau tableau :

```
bool  MathArctanh(
   const double&  array[],   // tableau des valeurs
   double&        result[]   // tableau des résultats
  )
```

Version retournant les résultats dans le tableau d'origine :

```
bool  MathArctanh(
   double&        array[]    // tableau des valeurs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

result[]

[out] Tableau des valeurs de sortie.

array[]

[out] Tableau des valeurs de sortie.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathSignif

Arrondit une valeur au nombre spécifié de chiffres dans la mantisse.

Version pour travailler avec une valeur réelle :

```
double  MathSignif(
   const double   x,          // valeur
   const int      digits      // nombre de chiffres décimaux
  )
```

Valeur de Retour

La valeur arrondie.

Version pour travailler avec un tableau de valeurs réelles retournant les résultats dans un autre tableau :

```
bool  MathSignif(
   const double&  array[],    // tableau des valeurs
   int            digits,     // nombre de chiffres décimaux
   double         result[]    // tableau des résultats
  )
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

Version pour travailler avec un tableau de valeurs réelles retournant les résultats dans le tableau d'origine :

```
bool  MathSignif(
   double&        array[],    // tableau des valeurs
   int            digits      // nombre de chiffres décimaux
  )
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

Paramètres

x

[in] Valeur réelle à arrondir.

digits

[in] Nombre de chiffres décimaux.

array[]

[in] Tableau de valeurs réelles.

array[]

[out] Tableau des valeurs de sortie.

result[]

© 2000-2024, MetaQuotes Ltd.

[out] Tableau des valeurs de sortie.

© 2000-2024, MetaQuotes Ltd.

MathRank

Calcule les rangs des éléments de tableau.

Version pour travailler avec un tableau de valeurs réelles :

```
bool  MathRank(
   const double&  array[],  // tableau des valeurs
   double&        rank[]    // tableau de rangs
  )
```

Version pour travailler avec un tableau de valeurs entières :

```
bool  MathRank(
   const int&     array[],  // tableau des valeurs
   double&        rank[]    // tableau de rangs
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

rank[]

[out] Tableau pour retourner les rangs.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCorrelationPearson

Calcule le coefficient de corrélation de Pearson.

Version pour travailler avec des tableaux de valeurs réelles :

```
bool  MathCorrelationPearson(
   const double&  array1[],  // le premier tableau de valeurs
   const double&  array2[],  // le deuxième tableau de valeurs
   double&        r          // coefficient de corrélation
  )
```

Version pour travailler avec des tableaux de valeurs entières :

```
bool  MathCorrelationPearson(
   const int&     array1[],  // le premier tableau de valeurs
   const int&     array2[],  // le deuxième tableau de valeurs
   double&        r          // coefficient de corrélation
  )
```

Paramètres

array1[]

[in] Le premier tableau de valeurs.

array2[]

[in] Le deuxième tableau de valeurs.

r

[out] Variable pour stocker le coefficient de corrélation.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCorrelationSpearman

Calcule le coefficient de corrélation de Spearman.

Version pour travailler avec des tableaux de valeurs réelles :

```
bool  MathCorrelationSpearman(
   const double&  array1[],  // le premier tableau de valeurs
   const double&  array2[],  // le deuxième tableau de valeurs
   double&        r          // coefficient de corrélation
  )
```

Version pour travailler avec des tableaux de valeurs entières :

```
bool  MathCorrelationSpearman(
   const int&     array1[],   // le premier tableau de valeurs
   const int&     array2[],   // le deuxième tableau de valeurs
   double&        r           // coefficient de corrélation
  )
```

Paramètres

array1[]

[in] Le premier tableau de valeurs.

array2[]

[in] Le deuxième tableau de valeurs.

r

[out] Variable pour stocker le coefficient de corrélation.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCorrelationKendall

Calcule le coefficient de corrélation de Kendall.

Version pour travailler avec des tableaux de valeurs réelles :

```
bool  MathCorrelationKendall(
   const double&  array1[],  // le premier tableau de valeurs
   const double&  array2[],  // le deuxième tableau de valeurs
   double&        tau        // coefficient de corrélation
  )
```

Version pour travailler avec des tableaux de valeurs entières :

```
bool  MathCorrelationKendall(
   const int&     array1[],  // le premier tableau de valeurs
   const int&     array2[],  // le deuxième tableau de valeurs
   double&        tau        // coefficient de corrélation
  )
```

Paramètres

array1[]

[in] Le premier tableau de valeurs.

array2[]

[in] Le deuxième tableau de valeurs.

tau

[out] Variable pour stocker le coefficient de corrélation.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathQuantile

Calcule des quantiles d'échantillons correspondant aux probabilités spécifiées. : Q[i](p) = (1 - gamma) *x[j] + gamma*x[j+1]

```
bool  MathQuantile(
   const double&  array[],     // tableau des valeurs
   const double&  probs[],     // tableau de probabilités
   double&        quantile[]   // tableau pour retourner les quantiles
  )
```

Paramètres

array[]

[in] Tableau des valeurs.

probs[]

[in] Tableau des probabilités.

quantile[]

[out] Tableau pour retourner les quantiles.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityEmpirical

La fonction calcule la fonction de densité de probabilité empirique (pdf (probability density function)) pour les valeurs aléatoires d'un tableau.

```
bool  MathProbabilityDensityEmpirical(
   const double&  array[],   // tableau de valeurs aléatoires
   const int      count,     // le nombre de paires
   double&        x[],       // tableau des valeurs x
   double&        pdf[]      // tableau de valeurs pdf
  )
```

Paramètres

array[]

[in] Tableau de valeurs aléatoires.

count

[in] Le nombre de paires (x, pdf(x)).

x[]

[out] Tableau pour retourner les valeurs x.

pdf[]

[out] Tableau pour retourner les valeurs pdf(x).

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionEmpirical

La fonction calcule la fonction de répartition cumulative empirique (cdf (cumulative distribution function)) pour les valeurs aléatoires d'un tableau.

```
bool  MathCumulativeDistributionEmpirical(
   const double&  array[],   // tableau de valeurs aléatoires
   const int      count,     // le nombre de paires
   double&        x[],       // tableau des valeurs x
   double&        cdf[]      // tableau de valeurs cdf
  )
```

Paramètres

array[]

[in] Tableau de valeurs aléatoires.

count

[in] Le nombre de paires (x, cdf(x)).

x[]

[out] Tableau pour retourner les valeurs x.

cdf[]

[out] Tableau pour retourner les valeurs de cdf(x).

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Fuzzy est une bibliothèque permettant de travailler avec de la logique floue

La **logique floue** est une synthèse de la logique aristotélicienne traditionnelle lorsque la vérité est marquée sous forme d'une variable linguistique. La logique floue, équivalente à la logique classique, possède ses propres opérations logiques floues sur des ensembles flous définis. Les mêmes opérations sur les ensembles ordinaires existent pour la logique floue, seuls leurs calculs sont, de loin, plus difficiles. Notez également que la composition d'ensembles flous constitue un autre ensemble flou.

Les grands principes de la logique floue, en la distinguant de la logique classique, sont la proximité maximale de la réflexion de la réalité et un haut niveau de subjectivité, ce qui peut mener à des erreurs significatives dans les calculs.

Le modèle (ou système) flou est un modèle mathématique dont les calculs sont basés sur la logique floue. Le construction de ce genre de modèle est applicable lorsque le sujet de l'étude a une formalisation faible et que sa description mathématique est trop complexe voire inconnue. La qualité des valeurs calculées de ces modèles (modèle d'erreur) ne dépend directement que de l'Expert Advisor ayant construit ce modèle. La meilleure option pour minimiser les erreurs est de dessiner le modèle le plus complet et le plus compréhensif et ensuite de l'ajuster en utilisant du machine learning sur un grand ensemble d'apprentissage.

L'avancée de la construction d'un modèle peut être divisée en trois étapes principales :

1. Définition des caractéristiques d'entrée et de sortie d'un modèle.
2. Construction de la base de connaissances.
3. Sélection de l'une des méthodes d'inférence floue (Mamdani ou Sugeno).

La première étape affecte directement les deux suivantes et détermine l'utilisation future du modèle. Une base de connaissances ou, comme on l'appelle parfois, une base de règles est un ensemble de règles floues : "si, alors" définissant la relation entre les entrées et les sorties de l'objet ex&miné. Le nombre de règles dans le système n'est pas limité et est déterminé par l'Expert Advisor. Le format général des régles floues est le suivant :

Si condition de la règle, alors conclusion de la règle.

La condition de la règle décrit l'état courant de l'objet, and la conclusion de la règle — comment cette condition affecte l'objet. Une vue générale des conditions et des conclusions n'est pas possible car elles sont déterminées par une inférence floue.

Chaque règle du système a son propre poids — cette caractéristique détermine l'importance d'une règle dans le modèle. Les facteurs de poids sont assignés à une règle dans l'intervalle [0, 1]. Dans de nombreux exemples sur les modèles flous, pouvant être trouvés dans la littérature correspondante, le poids n'est pas spécifié, mais cela ne signifie pas qu'il n'existe pas. De fait, dans ce cas chaque règle de la base se voit affectée un poids fixe de 1. Il peut y avoir deux types de termes et de conclusions pour chaque règle :

1. simple — inclut une seule variable floue;
2. complexe — inclut plusieurs variables floues.

Suivant la base de connaissances créée, le système d'inférence floue est déterminé par un modèle. Une inférence logique floue est un conteneur de conclusion sous la forme d'un ensemble flou correspondant

© 2000-2024, MetaQuotes Ltd.

à la valeur courante des entrées avec l'utilisation de la base de connaissances et des opérations floues. Les deux principaux types d'inférences floues sont Mamdani et Sugeno.

© 2000-2024, MetaQuotes Ltd.

Fonctions d'appartenance

Une **fonction d'appartenance** est une fonction permettant de calculer le degré d'appartenance d'un élément aléatoire de l'ensemble universel d'un ensemble flou. Le domaine d'une fonction d'appartenance doit donc être dans l'intervalle [0, 1].

Dans la plupart des cas, la fonction d'appartenance est continue et monotone.

Classes des fonctions d'appartenance	Description
CConstantMembershipFunction	Classe d'implémentation d'une fonction d'appartenance sous forme d'une ligne droite parallèle à l'axe des coordonnées
CCompositeMembershipFunction	Classe d'implémentation d'une composition de fonctions d'appartenance
CDifferencTwoSigmoidalMembershipFunction	Classe d'implémentation d'une fonction d'appartenance sous la forme d'une différence entre deux fonctions sigmoïdes avec les paramètres A1, A2, C1 et C2
CGeneralizedBellShapedMembershipFunction	Classe d'implémentation d'une fonction généralisée en forme de cloche avec les paramètres A, B et C
CNormalCombinationMembershipFunction	Classe d'implémentation d'une fonction d'appartenance double face avec les paramètres B1, B2, Sigma1 et Sigma2
CNormalMembershipFunction	Classe d'implémentation d'une fonction d'appartenance symétrique gaussienne avec les paramètres B et Sigma
CP_ShapedMembershipFunction	Classe d'implémentation d'une fonction d'appartenance en forme de pi avec les paramètres A, B, C et D
CProductTwoSigmoidalMembershipFunction	Classe d'implémentation d'une fonction d'appartenance sous la forme d'un produit de deux fonctions sigmoïdes avec les paramètres A1, A2, C1 et C2
CS_ShapedMembershipFunction	Classe d'implémentation d'une fonction d'appartenance en forme de S avec les paramètres A et B
CTrapezoidMembershipFunction	Classe d'implémentation d'une fonction d'appartenance trapézoïdale avec les paramètres X1, X2, X3 et X4
CTriangularMembershipFunction	Classe d'implémentation d'une fonction d'appartenance triangulaire avec les paramètres X1, X2 e X3

© 2000-2024, MetaQuotes Ltd.

Classes des fonctions d'appartenance	Description
CSigmoidalMembershipFunction	Classe d'implémentation d'une fonction d'appartenance sigmoïde avec les paramètres A et C
CZ_ShapedMembershipFunction	Classe d'implémentation d'une fonction d'appartenance en forme de Z avec les paramètres A et B.
IMembershipFunction	Classe de base de toutes les classes de fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

CConstantMembershipFunction

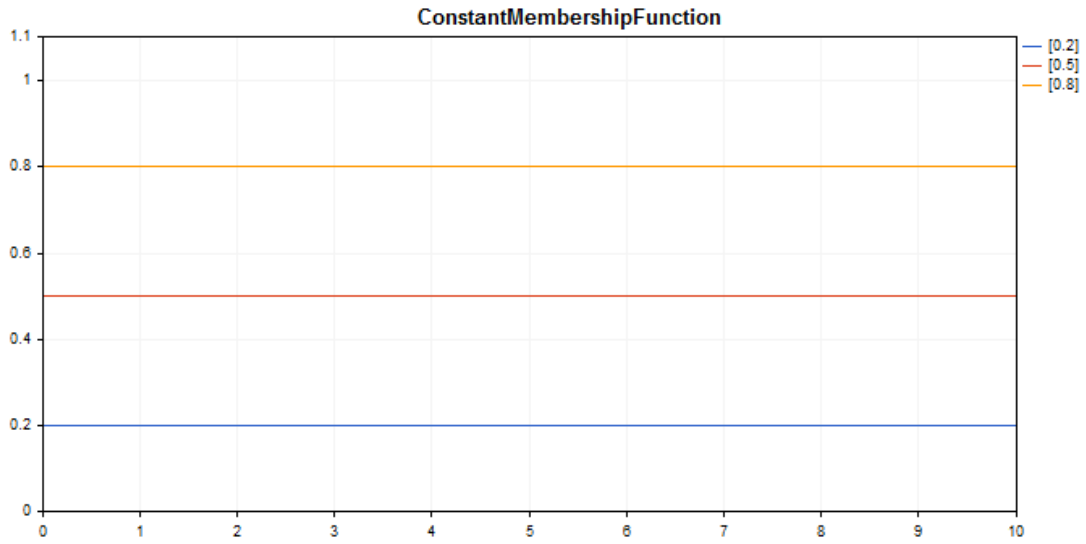
Classe d'implémentation d'une fonction d'appartenance sous forme d'une ligne droite parallèle à l'axe des coordonnées.

Description

La fonction est décrite avec l'équation :

y(x)=c

Le degré d'appartenance de la fonction est donc le même tout le long de l'axe numérique et est égal au paramètre spécifié dans le constructeur.



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CConstantMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CConstantMembershipFunction

Méthodes de classe

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                                   ConstantMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CConstantMembershipFunction func1(0.2);
CConstantMembershipFunction func2(0.5);
CConstantMembershipFunction func3(0.8);
//--- Crée les wrappers des fonctions membres
double ConstantMembershipFunction1(double x) { return(func1.GetValue(x)); }
double ConstantMembershipFunction2(double x) { return(func2.GetValue(x)); }
double ConstantMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"ConstantMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"ConstantMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ConstantMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(ConstantMembershipFunction1,0.0,10.0,1.0,CURVE_LINES,"[0.2]");
   graphic.CurveAdd(ConstantMembershipFunction2,0.0,10.0,1.0,CURVE_LINES,"[0.5]");
   graphic.CurveAdd(ConstantMembershipFunction3,0.0,10.0,1.0,CURVE_LINES,"[0.8]");
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
```

© 2000-2024, MetaQuotes Ltd.

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const double  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

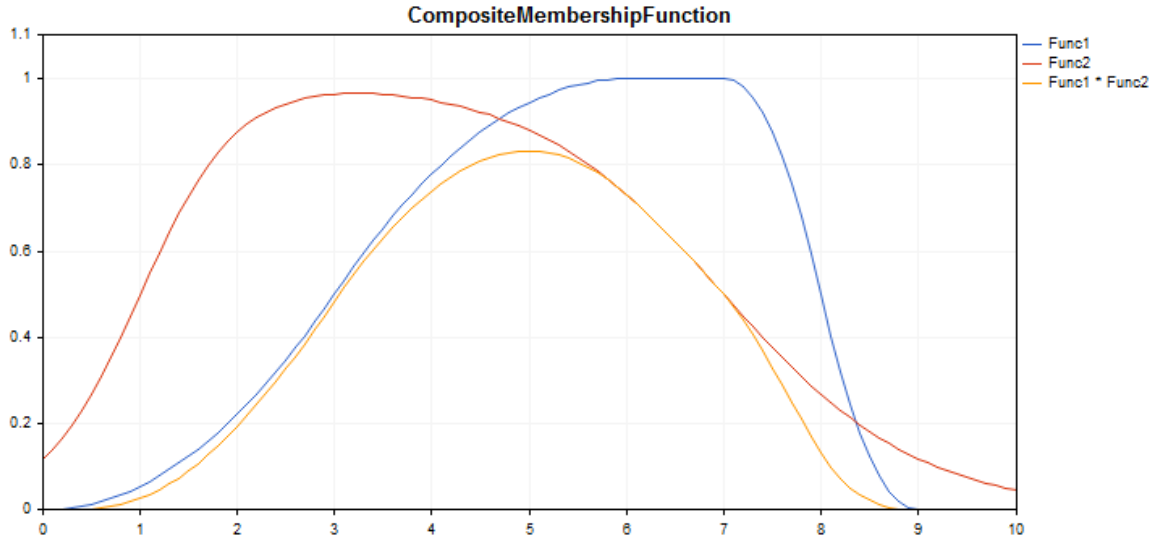
© 2000-2024, MetaQuotes Ltd.

CCompositeMembershipFunction

Classe d'implémentation d'une composition de fonctions d'appartenance.

Description

La composition de fonctions d'appartenance est une combinaison de deux, ou plus, fonctions d'appartenance utilisant un opérateur spécifié.



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CCompositeMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CCompositeMembershipFunction

Méthodes de classe

Méthode de classe	Description
CompositionType	Définit l'opérateur de la composition.
MembershipFunctions	Retourne la liste des fonctions d'appartenance.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                                  CompositeMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CP_ShapedMembershipFunction func2(0,6,7,9);
CCompositeMembershipFunction composite(ProdMF,GetPointer(func1),GetPointer(func2));
//--- Crée les wrappers des fonctions membres
double ProductTwoSigmoidalMembershipFunctions(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction(double x) { return(func2.GetValue(x)); }
double CompositeMembershipFunction(double x) { return(composite.GetValue(x)); }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"CompositeMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"CompositeMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("CompositeMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(P_ShapedMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1");
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions,0.0,10.0,0.1,CURVE_LINES,"F
   graphic.CurveAdd(CompositeMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1 * Func
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
```

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

CompositionType

Définit l'opérateur de la composition.

```
void  CompositionType(
   MfCompositionType  value      // type de l'opérateur
   )
```

Paramètres

value

[in] Type de l'opérateur de la composition.

Note

Les types suivants d'opérateur sont disponibles :

- MinMF (minimum des fonctions)
- MaxMF (maximum des fonctions)
- ProdMF (produit des fonctions)
- SumMF (somme des fonctions)

MembershipFunctions

Retourne la liste des fonctions d'appartenance de la composition.

```
CList*  MembershipFunctions(
   void        // liste des fonctions d'appartenance
   )
```

Valeur de Retour

Liste des fonctions d'appartenance.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

© 2000-2024, MetaQuotes Ltd.

Valeur de la fonction d'appartenance.

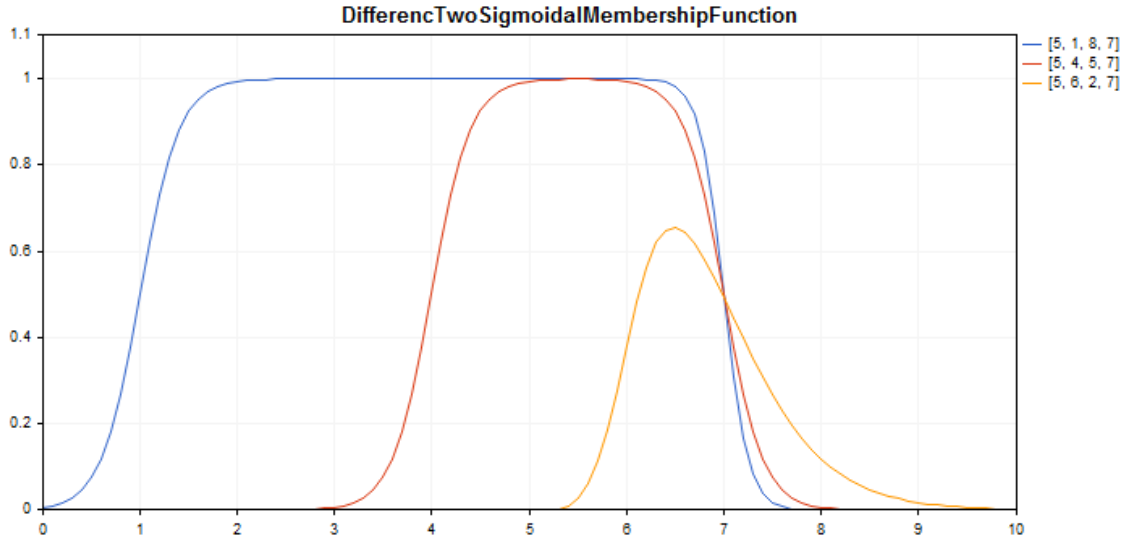
© 2000-2024, MetaQuotes Ltd.

CDifferencTwoSigmoidalMembershipFunction

Classe d'implémentation d'une fonction d'appartenance sous la forme d'une différence entre deux fonctions sigmoïdes avec les paramètres A1, A2, C1 et C2.

Description

La fonction est basée sur une courbe sigmoïde. Elle permet de créer des fonctions d'appartenance avec des valeurs égales à 1 commençant avec la valeur de l'argument. De telles fonctions conviennent si vous avez besoin de définir des termes linguistiques tels que "short" ou "long".



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CDifferencTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CDifferencTwoSigmoidalMembershipFunction

Méthodes de classe

Méthode de classe	Description
A1	Retourne et définit le rapport de la pente de la première fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
A2	Retourne et définit le rapport de la pente de la deuxième fonction d'appartenance.
C1	Retourne et définit le paramètre des coordonnées de l'inflexion de la première fonction d'appartenance.
C2	Retourne et définit le paramètre des coordonnées de l'inflexion de la deuxième fonction d'appartenance.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                      DifferencTwoSigmoidalMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CDifferencTwoSigmoidalMembershipFunction func1(5,1,8,7);
CDifferencTwoSigmoidalMembershipFunction func2(5,4,5,7);
CDifferencTwoSigmoidalMembershipFunction func3(5,6,2,7);
//--- Crée les wrappers des fonctions membres
double DifferencTwoSigmoidalMembershipFunction1(double x) { return(func1.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction2(double x) { return(func2.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"DifferencTwoSigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"DifferencTwoSigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("DifferencTwoSigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,
```

```
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (méthode Get)

Retourne le ratio de pente de la première fonction d'appartenance.

```
double  A1()
```

Valeur de Retour

Valeur du ratio de pente.

A1 (méthode Set)

Définit le ratio de pente de la première fonction d'appartenance.

```
void  A1(
   const double  a1      // valeur du ratio de pente
   )
```

Paramètres

a1

[in] Valeur du ratio de pente.

A2 (méthode Get)

Retourne le ratio de pente de la deuxième fonction d'appartenance.

```
double  A2()
```

Valeur de Retour

Valeur du ratio de pente.

A2 (méthode Set)

© 2000-2024, MetaQuotes Ltd.

Définit le ratio de pente de la deuxième fonction d'appartenance.

```
void  A2(
   const double  a2      // valeur du ratio de pente
   )
```

Paramètres

a2

[in] Valeur du ratio de pente.

C1 (méthode Get)

Retourne le paramètre de la coordonnée d'inflexion de la première fonction d'appartenance.

```
double  C1()
```

Valeur de Retour

Valeur des coordonnées d'inflexion.

C1 (méthode Set)

Définit le paramètre des coordonnées de l'inflexion de la première fonction d'appartenance.

```
void  C1(
   const double  c1      // valeur des coordonnées d'inflexion
   )
```

Paramètres

c1

[in] Valeur des coordonnées d'inflexion.

C2 (méthode Get)

Retourne le paramètre de coordonnée d'inflexion de la deuxième fonction d'appartenance.

```
double  C2()
```

Valeur de Retour

Valeur des coordonnées d'inflexion.

C2 (méthode Set)

Définit le paramètre des coordonnées de l'inflexion de la deuxième fonction d'appartenance.

```
void  C2(
   const double  c2      // valeur des coordonnées d'inflexion
   )
```

© 2000-2024, MetaQuotes Ltd.

Paramètres

c2

[in] Valeur des coordonnées d'inflexion.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const double  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

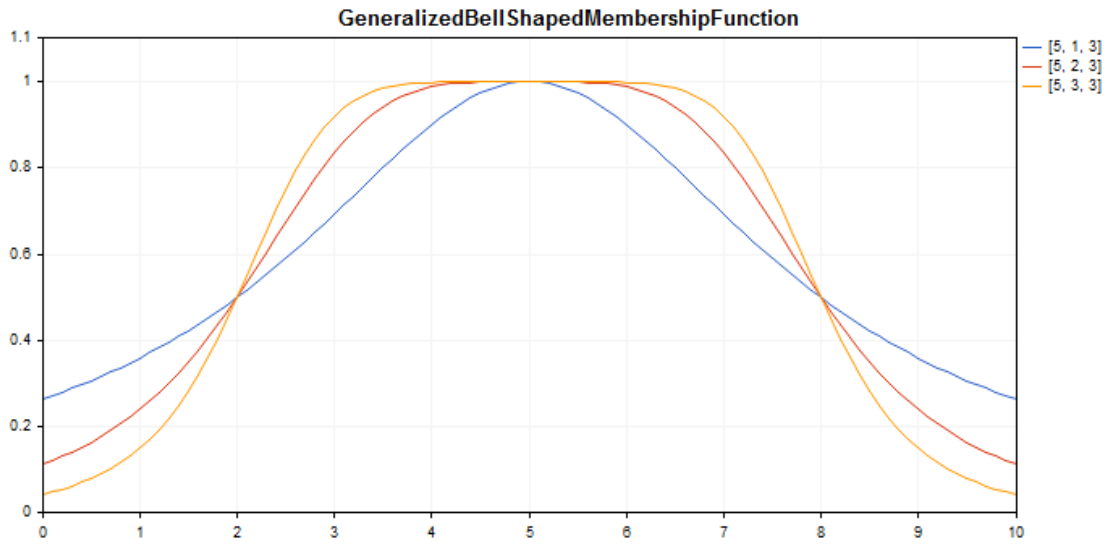
© 2000-2024, MetaQuotes Ltd.

CGeneralizedBellShapedMembershipFunction

Classe d'implémentation d'une fonction d'appartenance généralisée en forme de cloche avec les paramètres A, B et C.

Description

La forme d'une fonction d'appartenance en forme de cloche est similaire aux fonctions gaussiennes. La fonction est lisse et prend des valeurs différentes de zéro le long de toute la zone de définition.



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CGeneralizedBellShapedMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CGeneralizedBellShapedMembershipFunction

Méthodes de classe

Méthode de classe	Description
A	Retourne et définit le rapport de concentration de la fonction d'appartenance.
B	Retourne et définit le ratio de pente de la fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
C	Retourne et définit la coordonnée du maximum de la fonction d'appartenance.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                      GeneralizedBellShapedMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CGeneralizedBellShapedMembershipFunction func1(5, 1, 3);
CGeneralizedBellShapedMembershipFunction func2(5, 2, 3);
CGeneralizedBellShapedMembershipFunction func3(5, 3, 3);
//--- Crée les wrappers des fonctions membres
double GeneralizedBellShapedMembershipFunction1(double x) { return(func1.GetValue(x));
double GeneralizedBellShapedMembershipFunction2(double x) { return(func2.GetValue(x));
double GeneralizedBellShapedMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"GeneralizedBellShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"GeneralizedBellShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("GeneralizedBellShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (méthode Get)

Retourne le ratio de concentration de la fonction d'appartenance.

```
double  A()
```

Valeur de Retour

Valeur du ration de concentration.

A (méthode Set)

Définit le ratio de concentration de la fonction d'appartenance.

```
void  A(
   const double  a      // valeur du ratio de concentration
   )
```

Paramètres

a

[in] Ratio de concentration de la fonction d'appartenance.

B (méthode Get)

Retourne le ratio de pente de la fonction d'appartenance.

```
double  B()
```

Valeur de Retour

Valeur du ratio de pente.

B (méthode Set)

Définit le ratio de pente de la fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

```
void B(
   const double b      // valeur du ratio de pente
   )
```

Paramètres

b

[in] Ratio de pente de la fonction d'appartenance.

C (méthode Get)

Retourne la coordonnée du maximum de la fonction d'appartenance.

```
double C()
```

Valeur de Retour

la coordonnée du maximum de la fonction d'appartenance.

C (méthode Set)

Définit la coordonnée du maximum de la fonction d'appartenance.

```
void C(
   const double c      // valeur de la coordonnée du maximum
   )
```

Paramètres

c

[in] Coordonnée maximum de la fonction d'appartenance.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double GetValue(
   const x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

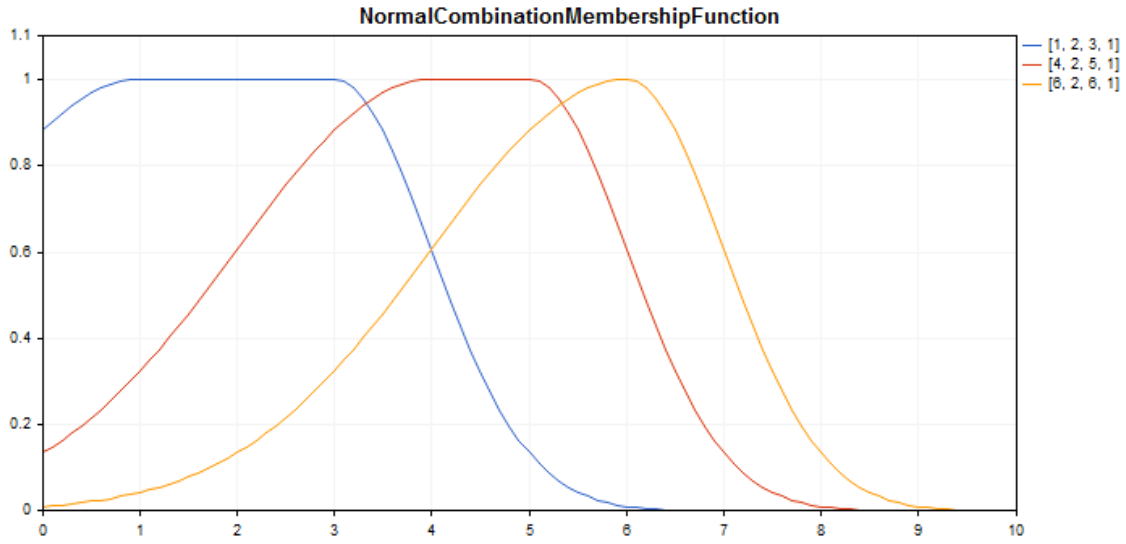
© 2000-2024, MetaQuotes Ltd.

CNormalCombinationMembershipFunction

Classe d'implémentation d'une fonction d'appartenance gaussienne double face avec les paramètres B1, B2, Sigma1 et Sigma2.

Description

La fonction d'appartenance gaussienne à double face est formée en utilisant une distribution gaussienne. Elle permet de définir des fonctions d'appartenance asymétriques. La fonction est lisse et prend des valeurs différentes de zéro le long de toute la zone de définition.



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CNormalCombinationMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
IMembershipFunction
CNormalCombinationMembershipFunction

Méthodes de classe

Méthode de classe	Description
B1	Retourne et définit la valeur du centre de la première fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
B2	Retourne et définit la valeur du centre de la deuxième fonction d'appartenance.
Sigma1	Retourne et définit le premier paramètre de la courbure de la fonction d'appartenance.
Sigma2	Retourne et définit le deuxième paramètre de la courbure de la fonction d'appartenance.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                          NormalCombinationMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CNormalCombinationMembershipFunction func1(1,2,3,1);
CNormalCombinationMembershipFunction func2(4,2,5,1);
CNormalCombinationMembershipFunction func3(6,2,6,1);
//--- Crée les wrappers des fonctions membres
double NormalCombinationMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalCombinationMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalCombinationMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"NormalCombinationMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalCombinationMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalCombinationMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(NormalCombinationMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[1,
```

```
   graphic.CurveAdd(NormalCombinationMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[4,
   graphic.CurveAdd(NormalCombinationMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[6,
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B1 (méthode Get)

Retourne la valeur du centre de la première fonction d'appartenance.

```
double  B1()
```

Valeur de Retour

Valeur du centre de la première fonction d'appartenance.

B1 (méthode Set)

Définit la valeur du centre de la première fonction d'appartenance.

```
void  B1(
   const double  b1      // valeur du premier centre
   )
```

Paramètres

b

[in] Valeur du centre de la première fonction d'appartenance.

B2 (méthode Get)

Retourne la valeur du centre de la deuxième fonction d'appartenance.

```
double  B2()
```

Valeur de Retour

La valeur du centre de la deuxième fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

B2 (méthode Set)

Définit la valeur du centre de la deuxième fonction d'appartenance.

```
void  B2(
   const double  b2      // valeur du deuxième centre
   )
```

Paramètres

b2

[in] Valeur du centre de la deuxième fonction d'appartenance.

Sigma1 (méthode Get)

Retourne le premier paramètre de courbure de la fonction d'appartenance.

```
double  Sigma1()
```

Valeur de Retour

Le premier paramètre de courbure de la fonction d'appartenance.

Sigma1 (méthode Set)

Définit la valeur du premier paramètre de courbure de la fonction d'appartenance.

```
void  Sigma1(
   const double  sigma1      // valeur du premier paramètre de courbure
   )
```

Paramètres

sigma1

[in] Le premier paramètre de courbure de la fonction d'appartenance.

Sigma2 (méthode Get)

Retourne le deuxième paramètre de courbure de la fonction d'appartenance.

```
double  Sigma2()
```

Valeur de Retour

La valeur du deuxième paramètre de courbure de la fonction d'appartenance.

Sigma2 (méthode Set)

© 2000-2024, MetaQuotes Ltd.

Définit la valeur du deuxième paramètre de courbure de la fonction d'appartenance.

```
void  Sigma2(
   const double  sigma2      // valeur de deuxième paramètre de courbure
   )
```

Paramètres

sigma2

[in] Le deuxième paramètre de courbure de la fonction d'appartenance.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

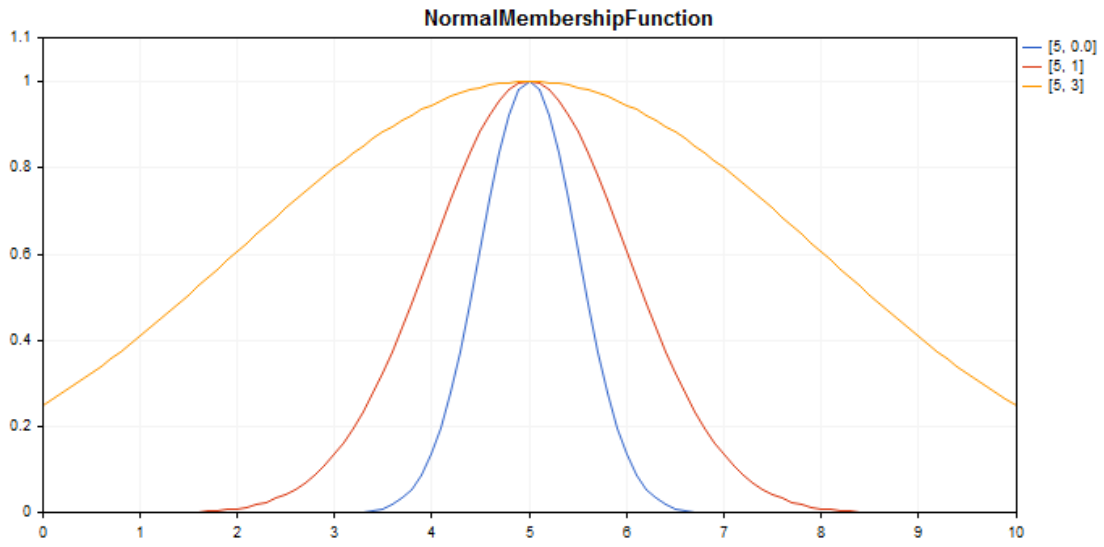
© 2000-2024, MetaQuotes Ltd.

CNormalMembershipFunction

Classe d'implémentation d'une fonction d'appartenance symétrique gaussienne avec les paramètres B et Sigma.

Description

La fonction d'appartenance gaussienne symétrique est formée en utilisant une distribution gaussienne. La fonction est lisse et prend des valeurs différentes de zéro le long de toute la zone de définition.



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CNormalMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CNormalMembershipFunction

Méthodes de classe

Méthode de classe	Description
B	Retourne et définit le centre de la fonction d'appartenance.
Sigma	Retourne et définit le paramètre de courbure de la fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                                   NormalMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CNormalMembershipFunction func1(5,0.5);
CNormalMembershipFunction func2(5,1);
CNormalMembershipFunction func3(5,3);
//--- Crée les wrappers des fonctions membres
double NormalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"NormalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(NormalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[5, 0.0]");
   graphic.CurveAdd(NormalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[5, 1]");
   graphic.CurveAdd(NormalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[5, 3]");
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
```

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B (méthode Get)

Retourne le centre de la fonction d'appartenance.

```
double  B()
```

Valeur de Retour

La valeur du centre de la fonction d'appartenance.

B (méthode Set)

Définit la valeur du centre de la fonction d'appartenance.

```
void  B(
   const double  b      // valeur du centre de la fonction
   )
```

Paramètres

b

[in] Valeur du centre de la fonction d'appartenance.

Sigma (méthode Get)

Retourne le paramètre de courbure de la fonction d'appartenance.

```
double  Sigma()
```

Valeur de Retour

La valeur du paramètre de courbure de la fonction d'appartenance.

Sigma (méthode Set)

Définit la valeur du paramètre de courbure de la fonction d'appartenance.

```
void  Sigma(
   const double  sigma      // valeur du paramètre de courbure
   )
```

Paramètres

sigma

© 2000-2024, MetaQuotes Ltd.

[in] Paramètre de courbure de la fonction d'appartenance.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const double  x      // argument
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

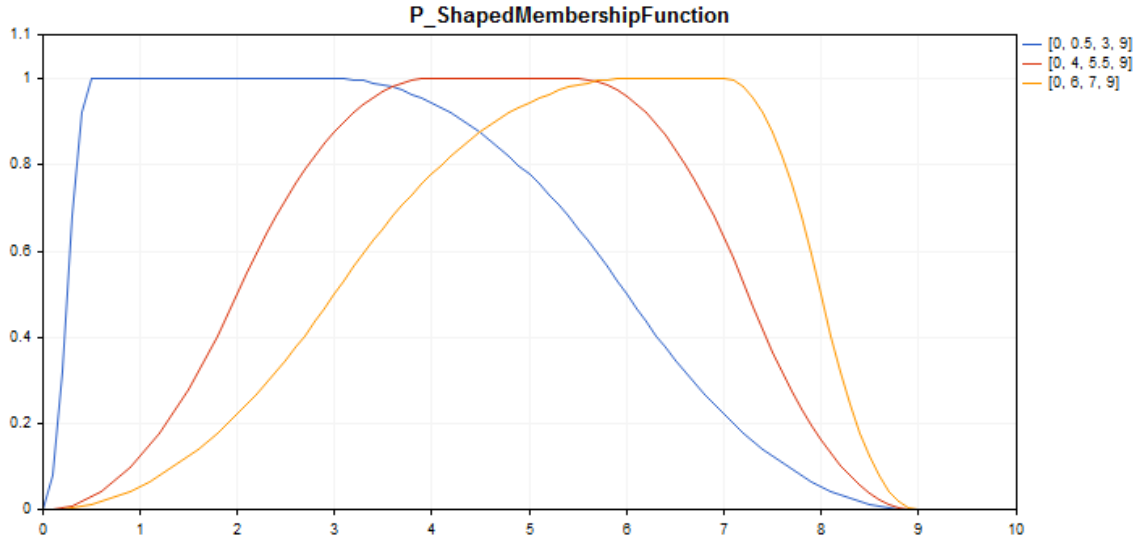
© 2000-2024, MetaQuotes Ltd.

CP_ShapedMembershipFunction

Classe d'implémentation d'une fonction d'appartenance en forme de pi avec les paramètres A, B, C et D.

Description

La fonction d'appartenance pi a la forme d'un trapèze curviligne. La fonction est utilisée pour définir des fonctions d'appartenance asymétriques ayant une transition en douceur d'une analyse pessimiste vers une analyse optimiste du nombre flou.



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CP_ShapedMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CP_ShapedMembershipFunction

Méthodes de classe

Méthode de classe	Description
A	Retourne et définit le paramètre de début de l'ensemble flou.

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
B	Retourne et définit le premier paramètre du coeur de l'ensemble flou.
C	Retourne et définit le deuxième paramètre du coeur de l'ensemble flou.
D	Retourne et définit le paramètre de fin de l'ensemble flou.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                                   P_ShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CP_ShapedMembershipFunction func1(0,0.5,3,9);
CP_ShapedMembershipFunction func2(0,4,5.5,9);
CP_ShapedMembershipFunction func3(0,6,7,9);
//--- Crée les wrappers des fonctions membres
double P_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double P_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"P_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"P_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("P_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(P_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 0.5, 3,
```

```
   graphic.CurveAdd(P_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 4, 5.5,
   graphic.CurveAdd(P_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[0, 6, 7, 9
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (méthode Get)

Retourne le paramètre de début de l'ensemble flou.

```
double  A()
```

Valeur de Retour

Le paramètre de début de l'ensemble flou.

A (méthode Set)

Définit le paramètre de début de l'ensemble flou.

```
void  A(
   const double  a      // paramètre de début de l'ensemble flou
   )
```

Paramètres

a

[in] Paramètre de début de l'ensemble flou.

B (méthode Get)

Retourne le premier paramètre du coeur de l'ensemble flou.

```
double  B()
```

Valeur de Retour

Le premier paramètre du coeur de l'ensemble flou.

B (méthode Set)

© 2000-2024, MetaQuotes Ltd.

Définit le premier paramètre du coeur de l'ensemble flou.

```
void B(
   const double b      // valeur du premier paramètre du coeur de l'ensemble flou
   )
```

Paramètres

b

[in] Le premier paramètre du coeur de l'ensemble flou.

C (méthode Get)

Retourne le deuxième paramètre du coeur de l'ensemble flou.

```
double C()
```

Valeur de Retour

Le deuxième paramètre du coeur de l'ensemble flou.

C (méthode Set)

Définit le deuxième paramètre du coeur de l'ensemble flou.

```
void C(
   const double c      // valeur du deuxième paramètre du coeur de l'ensemble flou
   )
```

Paramètres

c

[in] Le deuxième paramètre du coeur de l'ensemble flou.

D (méthode Get)

Retourne le paramètre de fin de l'ensemble flou.

```
double D()
```

Valeur de Retour

Valeur du paramètre de fin de l'ensemble flou.

D (méthode Set)

Définit le paramètre de fin de l'ensemble flou.

```
void D(
   const double d      // valeur du paramètre de fin de l'ensemble flou
   )
```

Paramètres

d

© 2000-2024, MetaQuotes Ltd.

[in] Valeur du paramètre de fin de l'ensemble flou.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const double  x
   )
```

Paramètres

x

[in] argument de la fonction d'appartenance

Valeur de Retour

valeur de la fonction d'appartenance

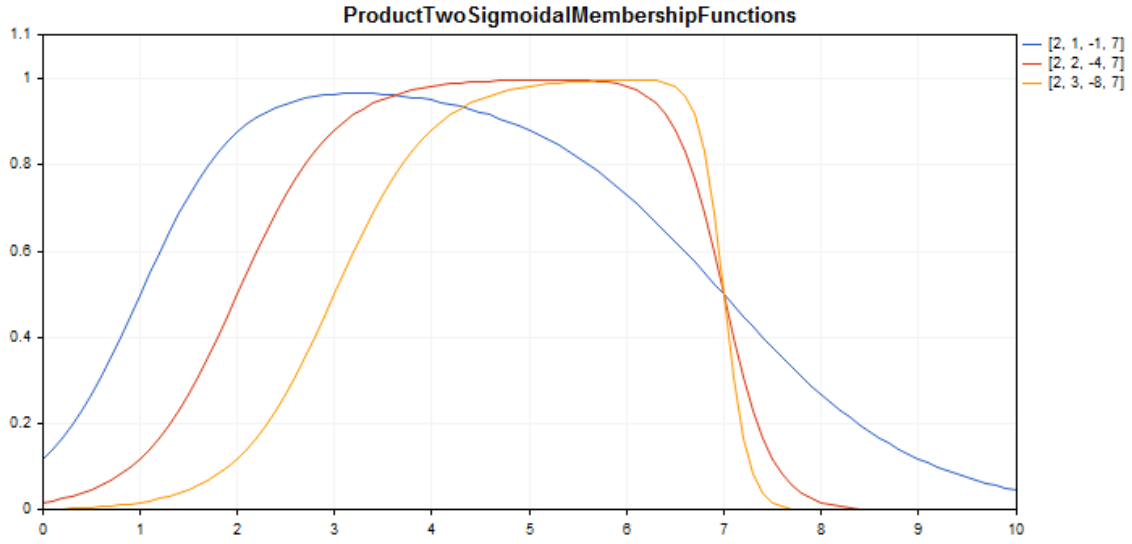
© 2000-2024, MetaQuotes Ltd.

CProductTwoSigmoidalMembershipFunction

Classe d'implémentation d'une fonction d'appartenance sous la forme d'un produit de deux fonctions sigmoïdes avec les paramètres A1, A2, C1 et C2.

Description

Le produit de deux fonctions d'appartenance sigmoïdes est appliqué pour définir les fonctions asymétriques lisses. Elle permet de créer des fonctions d'appartenance avec des valeurs égales à 1 commençant avec la valeur de l'argument. De telles fonctions conviennent si vous avez besoin de définir des termes linguistiques tels que "short" ou "long".



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CProductTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CProductTwoSigmoidalMembershipFunctions

Méthodes de classe

Méthode de classe	Description
A1	Retourne et définit le rapport de la pente de la première fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
A2	Retourne et définit le rapport de la pente de la deuxième fonction d'appartenance.
C1	Retourne le paramètre de la coordonnée d'inflexion de la première fonction d'appartenance.
C2	Retourne le paramètre de coordonnée d'inflexion de la deuxième fonction d'appartenance.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                      ProductTwoSigmoidalMembershipFunctions.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CProductTwoSigmoidalMembershipFunctions func2(2,2,-4,7);
CProductTwoSigmoidalMembershipFunctions func3(2,3,-8,7);
//--- Crée les wrappers des fonctions membres
double ProductTwoSigmoidalMembershipFunctions1(double x) { return(func1.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions2(double x) { return(func2.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"ProductTwoSigmoidalMembershipFunctions",0,30,30,780,380))
     {
      graphic.Attach(0,"ProductTwoSigmoidalMembershipFunctions");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ProductTwoSigmoidalMembershipFunctions");
```

```
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions1,0.0,10.0,0.1,CURVE_LINES,'
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions2,0.0,10.0,0.1,CURVE_LINES,'
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions3,0.0,10.0,0.1,CURVE_LINES,'
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (méthode Get)

Retourne le ratio de pente de la première fonction d'appartenance.

```
double  A1()
```

Valeur de Retour

Le ratio de pente de la première fonction d'appartenance.

A1 (méthode Set)

Définit le ratio de pente de la première fonction d'appartenance.

```
void  A1(
   const double  a1      // le ratio de pente de la première fonction d'appartenance
   )
```

Paramètres

a1

[in] Le ratio de pente de la première fonction d'appartenance.

A2 (méthode Get)

Retourne le ratio de pente de la deuxième fonction d'appartenance.

```
double  A2()
```

Valeur de Retour

Le ratio de pente de la deuxième fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

A2 (méthode Set)

Définit le ratio de pente de la deuxième fonction d'appartenance.

```
void  A2(
   const double  a2      // le ratio de pente de la deuxième fonction d'appartenance
   )
```

Paramètres

a2

[in] Le ratio de pente de la deuxième fonction d'appartenance.

C1 (méthode Get)

Retourne le paramètre de la coordonnée d'inflexion de la première fonction d'appartenance.

```
double  C1()
```

Valeur de Retour

La coordonnée d'inflexion de la première fonction d'appartenance.

C1 (méthode Set)

Définit la coordonnée d'inflexion de la première fonction d'appartenance.

```
void  C1(
   const double  c1      // la coordonnée d'inflexion de la première fonction d'appart
   )
```

Paramètres

c1

[in] La coordonnée d'inflexion de la première fonction d'appartenance.

C2 (méthode Get)

Retourne le paramètre de coordonnée d'inflexion de la deuxième fonction d'appartenance.

```
double  C2()
```

Valeur de Retour

La coordonnée d'inflexion de la deuxième fonction d'appartenance.

C2 (méthode Set)

Définit la coordonnée d'inflexion de la deuxième fonction d'appartenance.

```
void  C2(
   const double  c2      // la coordonnée d'inflexion de la deuxième fonction d'appart
   )
```

© 2000-2024, MetaQuotes Ltd.

Paramètres

c2

[in] La coordonnée d'inflexion de la deuxième fonction d'appartenance.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

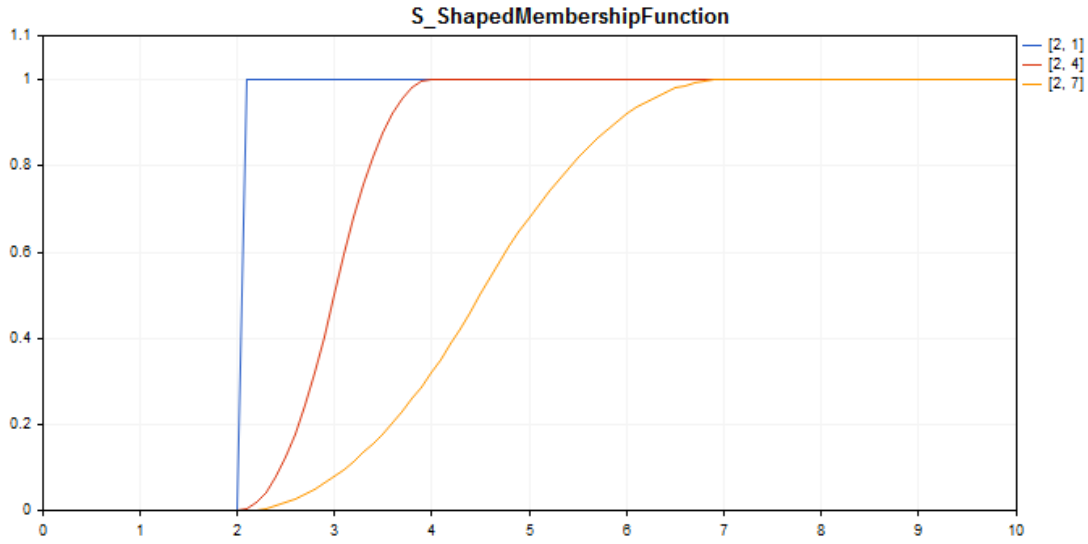
CS_ShapedMembershipFunction

Classe d'implémentation d'une fonction d'appartenance en forme de S avec les paramètres A et B.

Description

La fonction définit une fonction d'appartenance en forme de S et ayant deux paramètres. C'est une fonction non décroissante prenant des valeurs de 0 à 1. Les paramètres A et B définissent l'intervalle, dans lequel la fonction progresse dans une trajectoire non linéaire et de 0 à 1.

La fonction représente des ensembles flous de type "très haut" (c'est à dire que des fonctions d'appartenance non décroissantes avec saturation sont définies).



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CS_ShapedMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CS_ShapedMembershipFunction

Méthodes de classe

Méthode de classe	Description
A	Retourne et définit le paramètre du début de l'intervalle croissant.

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
B	Retourne et définit le premier paramètre du coeur de l'ensemble flou.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                                  S_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CS_ShapedMembershipFunction func1(2,1);
CS_ShapedMembershipFunction func2(2,4);
CS_ShapedMembershipFunction func3(2,7);
//--- Crée les wrappers des fonctions membres
double S_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double S_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double S_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"S_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"S_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("S_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(S_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(S_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
   graphic.CurveAdd(S_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 7]");
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (méthode Get)

Retourne le paramètre du début de l'intervalle croissant.

```
double  A()
```

Valeur de Retour

Paramètre du début de l'intervalle croissant.

A (méthode Set)

Définit le paramètre du début de l'intervalle croissant.

```
void  A(
   const double  a      // paramètre du début de l'intervalle croissant
   )
```

Paramètres

a

[in] Paramètre du début de l'intervalle croissant.

B (méthode Get)

Retourne le premier paramètre du coeur de l'ensemble flou.

```
double  B()
```

Valeur de Retour

Le premier paramètre du coeur de l'ensemble flou.

B (méthode Set)

Définit le premier paramètre du coeur de l'ensemble flou.

```
void  B(
   const double  b      // le premier paramètre du coeur de l'ensemble flou
   )
```

© 2000-2024, MetaQuotes Ltd.

Paramètres

b

[in] Le premier paramètre du coeur de l'ensemble flou.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

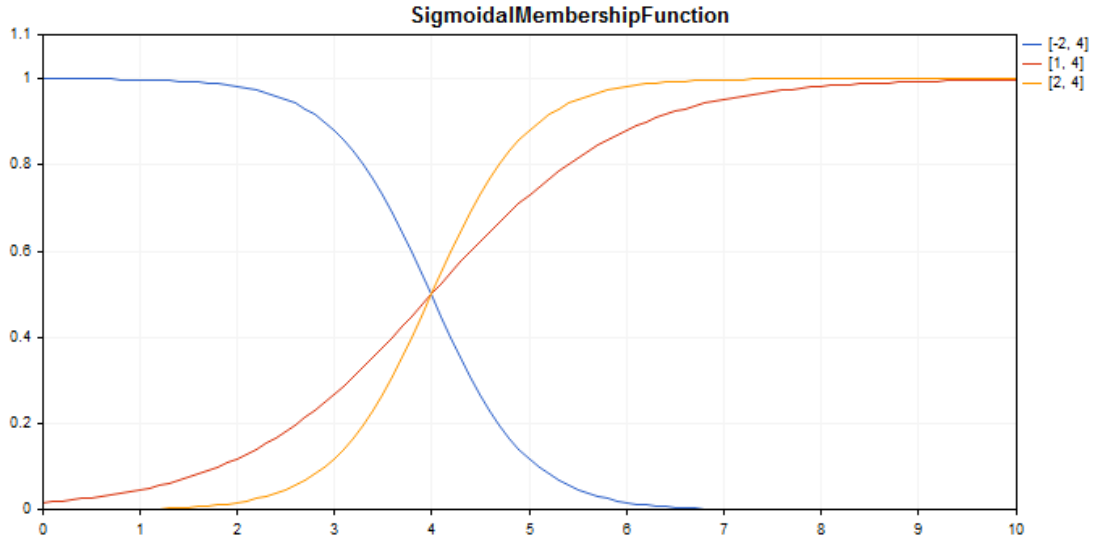
© 2000-2024, MetaQuotes Ltd.

CSigmoidalMembershipFunction

Classe d'implémentation d'une fonction d'appartenance sigmoïde avec les paramètres A et C.

Description

La fonction sigmoïde est appliquée lorsque l'on applique des fonctions d'appartenance monotones. Elle permet de créer des fonctions d'appartenance avec des valeurs égales à 1 commençant avec la valeur de l'argument. De telles fonctions conviennent si vous avez besoin de définir des termes linguistiques tels que "short" ou "long".



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CSigmoidalMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CSigmoidalMembershipFunction

Méthodes de classe

Méthode de classe	Description
A	Retourne et définit le ratio de pente de la fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
C	Retourne et définit le paramètre des coordonnées de l'inflexion de la fonction d'appartenance.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                                SigmoidalMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CSigmoidalMembershipFunction func1(-2, 4);
CSigmoidalMembershipFunction func2(1, 4);
CSigmoidalMembershipFunction func3(2, 4);
//--- Crée les wrappers des fonctions membres
double SigmoidalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double SigmoidalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double SigmoidalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"SigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"SigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("SigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(SigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[-2, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[1, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (méthode Get)

Retourne le ratio de pente de la fonction d'appartenance.

```
double  A()
```

Valeur de Retour

Le ratio de pente de la fonction d'appartenance.

A (méthode Set)

Définit le ratio de pente de la fonction d'appartenance.

```
void  A(
   const double  a      // le ratio de pente de la première fonction d'appartenance
   )
```

Paramètres

a

[in] Ratio de pente de la première fonction d'appartenance.

C (méthode Get)

Retourne la coordonnée d'inflexion de la fonction d'appartenance.

```
double  C()
```

Valeur de Retour

La coordonnée d'inflexion de la fonction d'appartenance.

C (méthode Set)

Définit la coordonnée d'inflexion de la fonction d'appartenance.

```
void  C(
   const double  c      // coordonnée d'inflexion de la fonction d'appartenance
   )
```

© 2000-2024, MetaQuotes Ltd.

Paramètres

c

[in] La coordonnée d'inflexion de la fonction d'appartenance.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

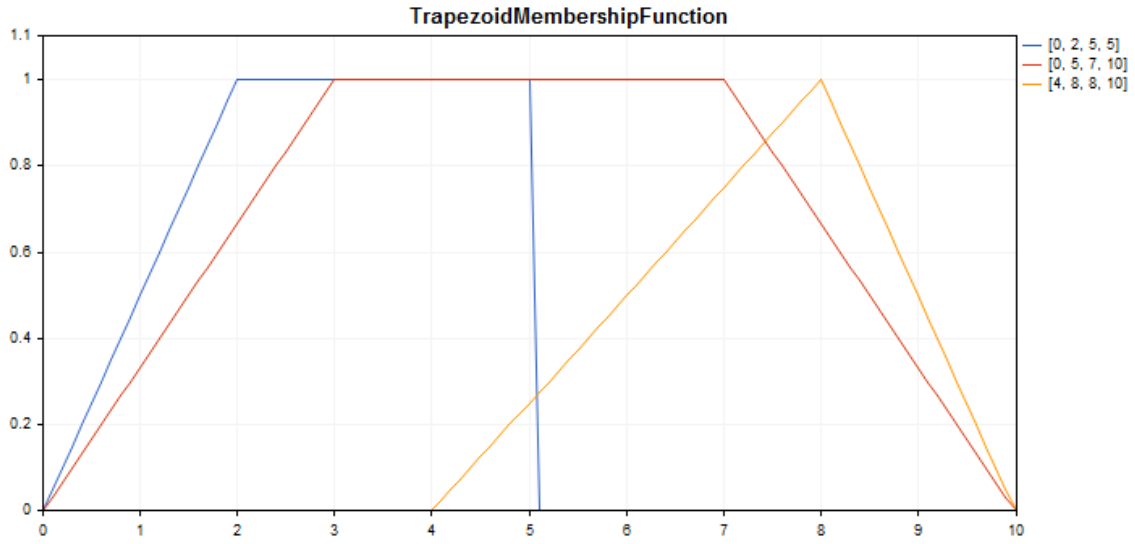
CTrapezoidMembershipFunction

Classe d'implémentation d'une fonction d'appartenance trapézoïdale avec les paramètres X1, X2, X3 et X4.

Description

La fonction est formée par approximation linéaire par morceaux. C'est une généralisation de la fonction triangulaire vous permettant d'assigner un noyau d'ensembles flous comme intervalle. Une telle fonction d'appartenance la rend possible pour interpréter de manière pratique des évaluations optimistes/pessimistes.

La fonction est utilisée pour définir des fonctions d'appartenance asymétriques des variables avec leurs valeurs les plus critiques définies dans un certain intervalle.



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CTrapezoidMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CTrapezoidMembershipFunction

Méthodes de classe

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
X1	Retourne et définit la valeur du premier point de l'axe X.
X2	Retourne et définit la valeur du deuxième point de l'axe X.
X3	Retourne et définit la valeur du troisième point de l'axe X.
X4	Retourne et définit la valeur du quatrième point de l'axe X.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                                TrapezoidMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CTrapezoidMembershipFunction func1(0,2,5,5);
CTrapezoidMembershipFunction func2(0,3,7,10);
CTrapezoidMembershipFunction func3(4,8,8,10);
//--- Crée les wrappers des fonctions membres
double TrapezoidMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TrapezoidMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TrapezoidMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"TrapezoidMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TrapezoidMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TrapezoidMembershipFunction");
   graphic.BackgroundMainSize(16);
```

```
//--- crée une courbe
   graphic.CurveAdd(TrapezoidMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5, 5
   graphic.CurveAdd(TrapezoidMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 7, 1
   graphic.CurveAdd(TrapezoidMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[4, 8, 8, 1
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (méthode Get)

Retourne la valeur du premier point de l'axe X.

```
double  X1()
```

Valeur de Retour

La valeur du premier point de l'axe X.

X1 (méthode Set)

Définit la valeur du premier point de l'axe X.

```
void  X1(
   const double  x1      // valeur du premier point de l'axe X
   )
```

Paramètres

x1

[in] La valeur du premier point de l'axe X.

X2 (méthode Get)

Retourne la valeur du deuxième point de l'axe X.

```
double  X2()
```

Valeur de Retour

La valeur du deuxième point de l'axe X.

© 2000-2024, MetaQuotes Ltd.

X2 (méthode Set)

Définit la valeur du deuxième point de l'axe X.

```
void  X2(
   const double  x2      // valeur du deuxième point de l'axe X
   )
```

Paramètres

x2

[in] La valeur du deuxième point de l'axe X.

X3 (méthode Get)

Retourne la valeur du troisième point de l'axe X.

```
double  X3()
```

Valeur de Retour

La valeur du troisième point de l'axe X.

X3 (méthode Set)

Définit la valeur du troisième point de l'axe X.

```
void  X3(
   const double  x3      // valeur du troisième point de l'axe X
   )
```

Paramètres

x3

[in] La valeur du troisième point de l'axe X.

X4 (méthode Get)

Retourne la valeur du quatrième point de l'axe X.

```
double  X4()
```

Valeur de Retour

La valeur du quatrième point de l'axe X.

X4 (méthode Set)

Définit la valeur du quatrième point de l'axe X.

```
void  X4(
   const double  x4      // valeur du quatrième point de l'axe X
   )
```

© 2000-2024, MetaQuotes Ltd.

Paramètres

x4

[in] La valeur du quatrième point de l'axe X.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

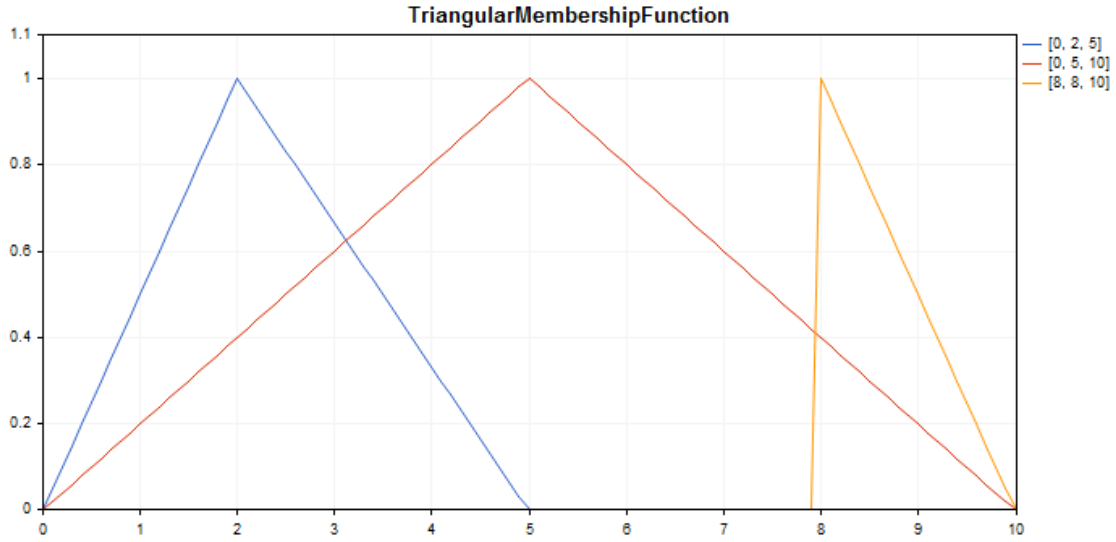
© 2000-2024, MetaQuotes Ltd.

CTriangularMembershipFunction

Classe d'implémentation d'une fonction d'appartenance triangulaire avec les paramètres X1, X2 et X3

Description

La fonction définit une fonction d'appartenance de forme triangulaire. C'est une fonction d'appartenance simple et est la plus fréquemment appliquée.



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CTriangularMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CTriangularMembershipFunction

Méthodes de classe

Méthode de classe	Description
X1	Retourne la valeur du premier point de l'axe X.
X2	Retourne la valeur du deuxième point de l'axe X.
X3	Retourne la valeur du troisième point de l'axe X.

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
ToNormalMF	Convertit une fonction d'appartenance triangulaire en une fonction gaussienne.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                                TriangularMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CTriangularMembershipFunction func1(0,2,5);
CTriangularMembershipFunction func2(0,5,10);
CTriangularMembershipFunction func3(8,8,10);
//--- Crée les wrappers des fonctions membres
double TriangularMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TriangularMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TriangularMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"TriangularMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TriangularMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TriangularMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(TriangularMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5]"
   graphic.CurveAdd(TriangularMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 10]
   graphic.CurveAdd(TriangularMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[8, 8, 10]
```

```
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (méthode Get)

Retourne la valeur du premier point de l'axe X.

```
double  X1()
```

Valeur de Retour

La valeur du premier point de l'axe X.

X1 (méthode Set)

Définit la valeur du premier point de l'axe X.

```
void  X1(
   const double  x1      // valeur du premier point de l'axe X
   )
```

Paramètres

x1

[in] La valeur du premier point de l'axe X.

X2 (méthode Get)

Retourne la valeur du deuxième point de l'axe X.

```
double  X2()
```

Valeur de Retour

La valeur du deuxième point de l'axe X.

X2 (méthode Set)

Définit la valeur du deuxième point de l'axe X.

```
void  X2(
```

© 2000-2024, MetaQuotes Ltd.

```
   const double  x2      // valeur du deuxième point de l'axe X
   )
```

Paramètres

x2

[in] La valeur du deuxième point de l'axe X.

X3 (méthode Get)

Retourne la valeur du troisième point de l'axe X.

```
double  X3()
```

Valeur de Retour

La valeur du troisième point de l'axe X.

X3 (méthode Set)

Définit la valeur du troisième point de l'axe X.

```
void  X3(
   const double  x3      // valeur du troisième point de l'axe X
   )
```

Paramètres

x3

[in] La valeur du troisième point de l'axe X.

ToNormalMF

Convertit une fonction d'appartenance triangulaire en une fonction gaussienne.

```
CNormalMembershipFunction*  ToNormalMF()
```

Valeur de Retour

Le pointeur vers une fonction d'appartenance gaussienne.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

© 2000-2024, MetaQuotes Ltd.

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

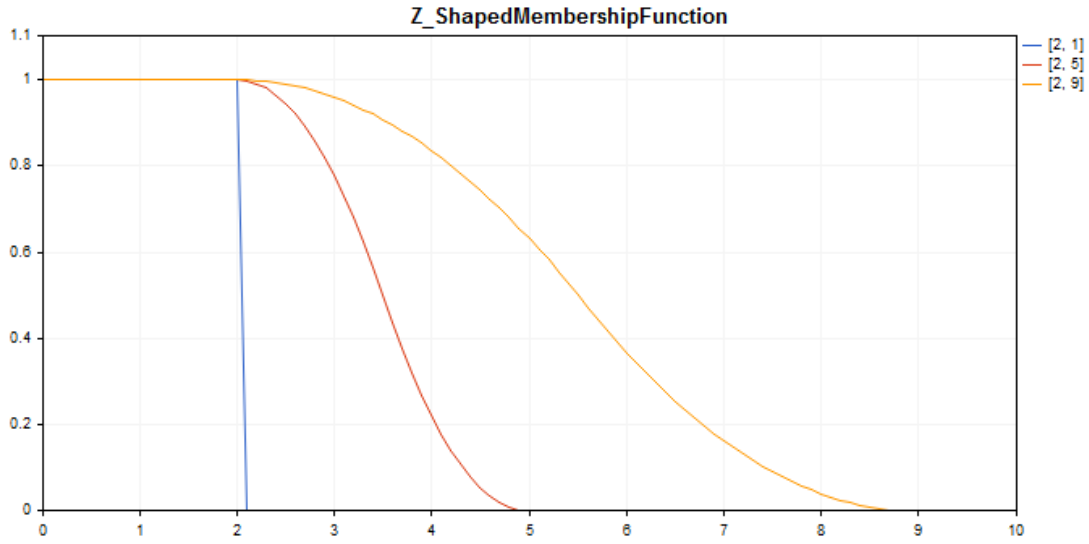
CZ_ShapedMembershipFunction

Classe d'implémentation d'une fonction d'appartenance en forme de Z avec les paramètres A et B.

Description

La fonction définit une fonction d'appartenance en forme de Z et ayant deux paramètres. C'est une fonction non croissante prenant des valeurs de 1 à 0. Les paramètres de la fonction définissent l'intervalle, dans lequel la fonction baisse dans une trajectoire non linéaire et de 1 à 0.

La fonction représente des ensembles flous de type "très bas". En d'autres termes, elle définit des fonctions d'appartenance non croissante avec saturation.



Un extrait de code dessinant un graphique est affiché plus bas.

Déclaration

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject
  IMembershipFunction
    CZ_ShapedMembershipFunction

Méthodes de classe

Méthode de classe	Description
A	Retourne et définit le paramètre du début de l'intervalle décroissant.

© 2000-2024, MetaQuotes Ltd.

Méthode de classe	Description
B	Retourne et définit le paramètre de fin de l'intervalle décroissant.
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemple

```
//+------------------------------------------------------------------+
//|                                  Z_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crée les fonctions membres
CZ_ShapedMembershipFunction func1(2,1);
CZ_ShapedMembershipFunction func2(2,5);
CZ_ShapedMembershipFunction func3(2,9);
//--- Crée les wrappers des fonctions membres
double Z_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double Z_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double Z_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"Z_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"Z_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("Z_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crée une courbe
   graphic.CurveAdd(Z_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(Z_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 5]");
   graphic.CurveAdd(Z_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 9]");
//--- définit les propriétés de l'axe X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- définit les propriétés de l'axe Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (méthode Get)

Retourne le paramètre du début de l'intervalle décroissant.

```
double  A()
```

Valeur de Retour

Paramètre du début de l'intervalle décroissant.

A (méthode Set)

Définit le paramètre du début de l'intervalle décroissant.

```
void  A(
   const double  a      // paramètre du début de l'intervalle décroissant
   )
```

Paramètres

a

[in] Paramètre du début de l'intervalle décroissant.

B (méthode Get)

Retourne le paramètre de fin de l'intervalle décroissant.

```
double  B()
```

Valeur de Retour

Paramètre de fin de l'intervalle décroissant.

B (méthode Set)

Définit le paramètre de fin de l'intervalle décroissant.

```
void  B(
   const double  b      // paramètre de fin de l'intervalle décroissant
   )
```

© 2000-2024, MetaQuotes Ltd.

Paramètres

b

[in] Paramètre de fin de l'intervalle décroissant.

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

IMembershipFunction

Classe de base de toutes les classes de fonction d'appartenance.

Déclaration

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Titre

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hiérarchie d'héritage

CObject

IMembershipFunction

Descendants directs

CCompositeMembershipFunction, CConstantMembershipFunction, CDifferencTwoSigmoidalMembershipFunction, CGeneralizedBellShapedMembershipFunction, CNormalCombinationMembershipFunction, CNormalMembershipFunction, CP_ShapedMembershipFunction, CProductTwoSigmoidalMembershipFunctions, CS_ShapedMembershipFunction, CSigmoidalMembershipFunction, CTrapezoidMembershipFunction, CTriangularMembershipFunction, CZ_ShapedMembershipFunction

Méthodes de classe

Méthode de classe	Description
GetValue	Calcule la valeur de la fonction d'appartenance d'un argument donné.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

GetValue

Calcule la valeur de la fonction d'appartenance d'un argument donné.

```
double  GetValue(
   const  x      // argument de fonction d'appartenance
   )
```

Paramètres

x

[in] Argument de fonction d'appartenance.

Valeur de Retour

Valeur de la fonction d'appartenance.

© 2000-2024, MetaQuotes Ltd.

Règles de systèmes flous

Un **système flou** (système d'inférence logique flou) est un conteneur de conclusions sous la forme d'un ensemble flou correspondant aux valeurs courantes des entrées avec l'utilisation d'un ensemble de **règles floues** et d'opérations floues.

Les **règles floues** déterminent la relation entre les entrées et les sorties d'un objet examiné. La quantité de règles dans le système est illimitée. La forme générale des règles floues est la suivante :

si *condition de la règle*, **alors** *conclusion de la règle*.

La *condition de la règle* décrit l'état actuel de l'objet. La *condition de la règle* décrit la façon dont la condition affecte l'objet.

Classe de règles pour les systèmes flous	Description
CMamdaniFuzzyRule	Classe d'implémentation d'une règle de logique floue pour l'algorithme Mamdani
CSugenoFuzzyRule	Classe d'implémentation d'une règle de logique floue pour l'algorithme Sugeno
CSingleCondition	La classe définit une condition floue exprimée par une paire "Variable floue — Terme flou".
CConditions	Classe définissant un ensemble de conditions floues connectées les unes aux autres par un opérateur.
CGenericFuzzyRule	Classe de base d'implémentation des deux types de règles floues.

© 2000-2024, MetaQuotes Ltd.

CMamdaniFuzzyRule

Inférence floue de type Mamdani — l'un des deux types de base des systèmes flous. Les valeurs des variables de sortie sont définies en utilisant des termes flous.

Description

La règle de logique floue pour l'algorithme de Mamdani peut être décrit de la façon suivante :

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

avec :

- X = (X1, X2, X3 ... Xn) — vecteur des variables d'entrée ;
- Y — variable de sortie ;
- a = (a1, a2, a3 ... an) — vecteur des valeurs des variables d'entrée ;
- d — valeur de la variable de sortie ;
- W — poids de la règle.

Déclaration

```
class CMamdaniFuzzyRule : public CGenericFuzzyRule
```

Titre

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Hiérarchie d'héritage

CObject
  IParsableRule
    CGenericFuzzyRule
      CMamdaniFuzzyRule

Méthodes de classe

Méthode de classe	Description
Conclusion	Retourne et définit la conclusion de la règle floue de Mamdani
Weight	Retourne et défini le poids de la règle floue de Mamdani

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

© 2000-2024, MetaQuotes Ltd.

Conclusion (méthode Get)

Retourne la conclusion de la règle floue de Mamdani

```
CSingleConditon*  Conclusion()
```

Valeur de Retour

Conclusion de la règle floue de Mamdani.

Conclusion (méthode Set)

Définit la conclusion de la règle floue de Mamdani.

```
void  Conclusion(
   CSingleConditon*  value      // conclusion de la règle floue de Mamdani
   )
```

Paramètres

value

[in] Conclusion d'une règle floue de Mamdani.

Weight (méthode Get)

Retourne le poids de la règle floue de Mamdani

```
double  Weight()
```

Valeur de Retour

Le poids de la règle floue de Mamdani.

Weight (méthode Set)

Définit le poids de la règle floue de Mamdani

```
void  Weight(
   const double  value      // poids de la règle floue de Mamdani
   )
```

Paramètres

value

[in] Poids de la règle floue de Mamdani.

© 2000-2024, MetaQuotes Ltd.

CSugenoFuzzyRule

Inférence floue de type Sugeno — l'un des deux types de base des systèmes flous. Les valeurs des variables de sortie sont définies comme une combinaison linéaire des variables d'entrée.

Description

Contrairement à la règle de Mamdani, la valeur d'une variable d'entrée est définie par une fonction linéaire à partir des entrées plutôt que par un terme flou. La règle de logique floue pour l'algorithme de Sugeno peut être décrit de la façon suivante :

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

avec :

- X = (X1, X2, X3 ... Xn) — vecteur des variables d'entrée ;
- Y — variable de sortie ;
- a = (a1, a2, a3 ... an) — vecteur des valeurs des variables d'entrée ;
- b = (b1, b2, b3 ... bn) — ratio de terme libre dans la fonction linéaire pourune valeur de sortie
- W — poids de la règle.

Déclaration

```
class CSugenoFuzzyRule : public CGenericFuzzyRule
```

Titre

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Hiérarchie d'héritage

CObject
  IParsableRule
    CGenericFuzzyRule
      CSugenoFuzzyRule

Méthodes de classe

Méthode de classe	Description
Conclusion	Retourne et définit la conclusion de la règle floue de Sugeno

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

© 2000-2024, MetaQuotes Ltd.

Conclusion (méthode Get)

Retourne la conclusion de la règle floue de Sugeno.

```
CSingleConditon*  Conclusion()
```

Valeur de Retour

La conclusion de la règle floue de Sugeno.

Conclusion (méthode Set)

Définit la conclusion de la règle floue de Sugeno.

```
void  Conclusion(
   CSingleCondition*  value      // conclusion d'une règle floue de Sugeno
   )
```

Paramètres

value

[in] Conclusion d'une règle floue de Sugeno.

© 2000-2024, MetaQuotes Ltd.

CSingleCondition

La classe définit une condition floue exprimée par une paire "Variable floue — Terme flou".

Description

Selon une condition floue, une variable correspond à un terme. Une condition floue peut être décrite avec l'expression suivante : *X est a*,

avec :

- X est une variable floue ;
- a est une valeur de la variable floue (terme flou).

Déclaration

```
class CSingleCondition : public ICondition
```

Titre

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Hiérarchie d'héritage

CObject
  ICondition
    CSingleCondition

Descendants directs

CFuzzyCondition

Méthodes de classe

Méthode de classe	Description
Not	Retourne et définit le flag indiquant s'il est nécessaire d'appliquer la négation à cette condition.
Term	Retourne et définit un terme flou pour cette condition.
Var	Retourne et définit une variable floue pour cette condition.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Not (méthode Get)

Retourne le flag indiquant s'il est nécessaire d'appliquer la négation à cette condition.

```
bool Not()
```

© 2000-2024, MetaQuotes Ltd.

Valeur de Retour

La valeur du flag.

Not (méthode Set)

Définit le flag indiquant s'il est nécessaire d'appliquer la négation à cette condition.

```
void  Not(
   bool  not      // valeur du flag
   )
```

Paramètres

not

[in] La valeur du flag.

Term (méthode Get)

Retourne un terme flou pour la condition donnée.

```
INamedValue* Term()
```

Valeur de Retour

Un terme flou pour la condition donnée.

Term (méthode Set)

Définit un terme flou pour la condition donnée.

```
void  Term(
   INamedValue*&  value      // terme flou pour la condition donnée
   )
```

Paramètres

value

[in] Un terme flou pour la condition donnée.

Var (méthode Get)

Retourne une variable floue pour la condition donnée.

```
INamedVariable* Var()
```

Valeur de Retour

Une variable floue pour la condition donnée.

Var (méthode Set)

Définit une variable floue pour la condition donnée.

© 2000-2024, MetaQuotes Ltd.

```
void  Var(
   INamedVariable*&  value      // une variable floue pour la condition donnée
   )
```

Paramètres

value

[in] Variable floue.

© 2000-2024, MetaQuotes Ltd.

CConditions

Classe définissant un ensemble de conditions floues connectées les unes aux autres par un opérateur.

Description

Un ensemble de conditions floues connectées les unes aux autres par un opérateur peut être décrit comme suit :

$$(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)$$

avec :

- X = (X1, X2, X3 ... Xn) — vecteur des variables d'entrée ;
- a = (a1, a2, a3 ... an) — vecteur des valeurs des variables d'entrée.

Dans cet exemple, l'opérateur *and* est utilisé. En outre, l'opérateur *or* est disponible dans cette classe.

Déclaration

```
class CConditions : public ICondition
```

Titre

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Hiérarchie d'héritage

CObject
  ICondition
    CConditions

Méthodes de classe

Méthode de classe	Description
ConditionsList	Retourne la liste de toutes les conditions
Not	Retourne et définit le flag indiquant s'il est nécessaire d'appliquer la négation à ces conditions
Op	Retourne et définit le type de l'opérateur des ensembles de conditions.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

ConditionsList

Retourne la liste de toutes les conditions.

```
CList*  ConditionsList()
```

© 2000-2024, MetaQuotes Ltd.

Valeur de Retour

Liste de toutes les conditions.

Not (méthode Get)

Retourne le flag indiquant s'il est nécessaire d'appliquer la négation à ces conditions.

```
bool Not()
```

Valeur de Retour

La valeur du flag.

Not (méthode Set)

Définit le flag indiquant s'il est nécessaire d'appliquer la négation à ces conditions.

```
void  Not(
   bool  not      // valeur du flag
   )
```

Paramètres

not

[in] La valeur du flag.

Op (méthode Get)

Retourne le type de l'opérateur de l'ensemble des conditions. Les opérateurs *and* et *or* sont disponibles.

```
OperatorType Op()
```

Valeur de Retour

Type de l'opérateur de l'ensemble des conditions.

Op (méthode Set)

Définit l'opérateur de l'ensemble des conditions. Les opérateurs *and* et *or* sont disponibles.

```
void  Op(
   OperatorType  op     // type de l'opérateur de l'ensemble des conditions
   )
```

Paramètres

op

[in] Type de l'opérateur de l'ensemble des conditions.

© 2000-2024, MetaQuotes Ltd.

CGenericFuzzyRule

Classe de base des deux types de règles floues.

Déclaration

```
class CGenericFuzzyRule : public IParsableRule
```

Titre

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Hiérarchie d'héritage

CObject
IParsableRule
CGenericFuzzyRule

Descendants directs

CMamdaniFuzzyRule, CSugenoFuzzyRule

Méthodes de classe

Méthode de classe	Description
Conclusion	Retourne et définit la conclusion d'une règle floue
Condition	Retourne et définit la condition 'if' (ensemble de conditions) pour une règle floue
CreateCondition	Crée une condition pour une règle floue avec les paramètres spécifiés

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Conclusion (méthode Get)

Retourne la conclusion d'une règle floue.

```
CSingleConditon*  Conclusion()
```

Valeur de Retour

Conclusion d'une règle floue.

Conclusion (méthode Set)

Définie la conclusion de la règle floue.

```
virtual void  Conclusion(
   CSingleConditon*  value      // conclusion d'une règle floue
```

© 2000-2024, MetaQuotes Ltd.

```
    )
```

Paramètres

value

[in] Conclusion d'une règle floue.

Condition (méthode Get)

Retourne la condition 'if' (ensemble de conditions) d'une règle floue.

```
CConditons*  Condition()
```

Valeur de Retour

Condition floue (ensemble de conditions).

Condition (méthode Set)

Définit la condition 'if' (ensemble de conditions) d'une règle floue.

```
void  Condition(
   CConditons*  value      // condition 'if' (ensemble de conditions) pour une règle f
   )
```

Paramètres

value

[in] Condition floue (ensemble de conditions).

CreateCondition

Crée une condition pour une règle floue avec les paramètres spécifiés.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,       // variable floue
   CFuzzyTerm*      term,      // terme flou
   )
```

Paramètres

var

[in] Variable floue.

term

[in] Terme flou.

Valeur de Retour

Statut de la règle floue.

CreateCondition

© 2000-2024, MetaQuotes Ltd.

Crée une condition pour une règle floue avec les paramètres spécifiés.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,       // variable floue
   CFuzzyTerm*      term,      // terme flou
   bool             not,       // flag indiquant s'il est nécessaire d'appliquer la néc
   )
```

Paramètres

var

[in] Variable floue.

term

[in] Terme flou.

not

[in] Flag indiquant s'il est nécessaire d'appliquer la négation à une condition.

Valeur de Retour

Statut de la règle floue.

CreateCondition

Crée une condition pour une règle floue avec les paramètres spécifiés.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable* var,       // variable floue
   CFuzzyTerm*     term,      // terme flou
   bool            not,       // flag indiquant s'il est nécessaire d'appliquer la néga
   HedgeType       hedge      // type de l'ensemble de conditions
   )
```

Paramètres

var

[in] Variable floue.

term

[in] Terme flou.

not

[in] Flag indiquant s'il est nécessaire d'appliquer la négation à une condition.

hedge

[in] Type de l'ensemble de conditions.

Valeur de Retour

Statut de la règle floue.

© 2000-2024, MetaQuotes Ltd.

Variables de systèmes flous

Les variables (linguistiques) floues **sont appliquées dans les systèmes flous**. Ce sont les variables dont les valeurs sont des mots ou des combinaisons de mots dans un langage naturel ou articiel.

Les variables linguistiques comprennent les ensembles flous. La nature et le nombre de variables floues changent pour chaque tâche lors de la définition des ensembles flous.

Classe	Description
CFuzzyVariable	Classe de création de variables floues génériques.
CSugenoVariable	Classe de création de variables floues du type Sugeno.

© 2000-2024, MetaQuotes Ltd.

CFuzzyVariable

Classe de création de variables floues génériques.

Description

Ici, une variable floue est créée avec les paramètres suivants :

- valeur maximum de la variable ;
- valeur minimum de la variable ;
- nom de la variable floue ;
- ensemble de termes (ensemble de toutes les valeurs possibles qu'une variable linguistique peut reçevoir).

Déclaration

```
class CFuzzyVariable : public CNamedVariableImpl
```

Titre

```
#include <Math\Fuzzy\fuzzyvariable.mqh>
```

Hiérarchie d'héritage

CObject
  INamedValue
    INamedVariable
      CNamedVariableImpl
        CFuzzyVariable

Méthodes de classe

Méthode de classe	Description
AddTerm	Ajoute un terme flou simple à une variable floue.
GetTermByName	Retourne le terme flou ayant le nom spécifié.
Max	Retourne et définit la valeur maximum d'une variable floue.
Min	Retourne et définit la valeur minimum d'une variable floue.
Terms	Retourne et définit la liste des termes flous d'une variable floue donnée.
Values	Retourne et définit la liste des termes flous d'une variable floue donnée.

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CNamedVariableImpl

Name, Name

AddTerm

Ajoute un terme flou simple à une variable floue.

```
void  AddTerm(
   CFuzzyTerm*&  term      // terme flou
   )
```

Paramètres

term

[in] Terme flou.

GetTermByName

Retourne le terme flou ayant le nom spécifié.

```
CFuzzyTerm*  GetTermByName(
   const string  name      // nom du terme flou
   )
```

Paramètres

name

[in] Nom du terme flou.

Valeur de Retour

Terme flou ayant le nom spécifié.

Max (méthode Get)

Retourne la valeur maximum d'une variable floue.

```
double  Max()
```

Valeur de Retour

La valeur maximum d'une variable floue.

Max (méthode Set)

Définit la valeur maximum d'une variable floue.

```
void  Max(
   const double  max      // valeur maximum d'une variable floue
```

© 2000-2024, MetaQuotes Ltd.

```
    )
```

Paramètres

max

[in] Valeur maximum d'une variable floue.

Min (méthode Get)

Retourne la valeur minimum d'une variable floue.

```
double  Min()
```

Valeur de Retour

La valeur minimum d'une variable floue.

Max (méthode Set)

Définit la valeur minimum d'une variable floue.

```
void  Min(
   const double  min      // valeur minimum d'une variable floue
   )
```

Paramètres

min

[in] Valeur minimum d'une variable floue.

Terms (méthode Get)

Retourne la liste des termes flous d'une variable floue donnée.

```
CList*  Terms()
```

Valeur de Retour

La liste des termes flous d'une variable floue donnée.

Terms (méthode Set)

Définit la liste des termes flous d'une variable floue donnée.

```
void  Terms(
   CList*&  terms      // liste des termes flous de la variable donnée
   )
```

Paramètres

terms

[in] liste de termes flous d'une variable floue donnée.

© 2000-2024, MetaQuotes Ltd.

Values

Retourne la liste des termes flous d'une variable floue donnée.

```
CList*  Values()
```

Valeur de Retour

La liste des termes flous de la variable donnée.

© 2000-2024, MetaQuotes Ltd.

CSugenoVariable

Classe de création de variables floues du type Sugeno.

Description

Une variable floue de rype Sugeno est différente d'une variable linguistique générale puisqu'elle n'est pas définie par un ensemble de termes mais par un ensemble de fonctions linéaires.

Déclaration

```
class CSugenoVariable : public CNamedVariableImpl
```

Titre

```
#include <Math\Fuzzy\sugenovariable.mqh>
```

Hiérarchie d'héritage

CObject
  INamedValue
    INamedVariable
      CNamedVariableImpl
        CSugenoVariable

Méthodes de classe

Méthode de classe	Description
Functions	Retourne la liste des fonctions linéaires d'une variable floue de Sugeno.
GetFuncByName	Retourne la fonction linéaire ayant le nom spécifié.
Values	Retourne la liste des fonctions linéaires d'une variable floue de Sugeno.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CNamedVariableImpl

Name, Name

Functions

Retourne la liste des fonctions linéaires d'une variable floue de Sugeno.

```
CList*  Functions()
```

Valeur de Retour

© 2000-2024, MetaQuotes Ltd.

Liste des fonctions linéaires.

GetFuncByName

Retourne la fonction linéaire ayant le nom spécifié.

```
ISugenoFunction*  GetFuncByName(
   const string  name      // nom de la fonction linéaire
   )
```

Paramètres

name

[in] Nom de la fonction linéaire.

Valeur de Retour

Fonction linéaire ayant le nom spécifié.

Values

Retourne la liste des fonctions linéaires d'une variable floue de Sugeno.

```
CList*  Values()
```

Valeur de Retour

Liste des fonctions linéaires d'une variable floue de Sugeno.

© 2000-2024, MetaQuotes Ltd.

CFuzzyTerm (termes flous)

Classe d'implémentation des termes flous.

Description

Un terme est un élément d'un ensemble de termes. Un terme est définit par ses deux composants :

- nom du terme flou ;
- fonction d'appartenance.

Déclaration

```
class CFuzzyTerm : public CNamedValueImpl
```

Titre

```
#include <Math\Fuzzy\fuzzyterm.mqh>
```

Hiérarchie d'héritage

CObject
  INamedValue
    CNamedValueImpl
      CFuzzyTerm

Méthodes de classe

Méthode de classe	Description
MembershipFunction	Retourne la fonction d'appartenance du terme flou.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CNamedValueImpl

Name, Name

© 2000-2024, MetaQuotes Ltd.

MembershipFunction

Retourne la fonction d'appartenance du terme flou.

```
IMembershipFunction*  MembershipFunction()
```

Valeur de Retour

La fonction d'appartenance

© 2000-2024, MetaQuotes Ltd.

Systèmes flous

Un **système flou** (ou *modèle flou*) est un modèle mathématique dont le calcul est basé sur la logique floue. Le construction de ce genre de modèle est applicable lorsque le sujet de l'étude a une formalisation faible et que sa description mathématique est trop complexe voire inconnue.

L'avancée de la construction d'un modèle peut être divisée en trois étapes principales :

1. Définition des caractéristiques d'entrée et de sortie d'un modèle.
2. Construction de la base de connaissances.
3. Sélection de l'une des méthodes d'inférence floue (Mamdani et Sugeno).

La première étape affecte directement les deux suivantes et détermine l'utilisation future du modèle.

Une base de connaissances (*base de règles*) est un ensemble de règles floues du type "si, alors", définissant la relation entre les entrées et les sorties de l'objet examiné.

La **condition de la règle** décrit l'état actuel de l'objet et la **conclusion de la règle** — comment cette condition affecte l'objet.

Il peut y avoir deux types de termes et de conclusions pour chaque règle :

1. simple (Csinglcond) — inclut une seule variable floue ;
2. complexe (Cconditions) — inclut plusieurs variables floues.

Chaque règle du système a son propre **poids** — correspondant à l'importance d'une règle dans le modèle. Les facteurs de poids sont assignés à une règle dans l'intervalle [0, 1].

Suivant la base de connaissances créée, le système d'inférence floue est déterminé par un modèle. Une **inférence logique floue** est un conteneur de conclusion sous la forme d'un ensemble flou correspondant à la valeur courante des entrées avec l'utilisation de la base de connaissances et des opérations floues. Les deux principaux types d'inférences floues sont Mamdani et Sugeno.

© 2000-2024, MetaQuotes Ltd.

Système de Mamdani

Les valeurs des variables de sortie du système de Mamdani sont définies en utilisant des termes flous.

Description

La règle de logique floue pour l'algorithme de Mamdani peut être décrit de la façon suivante :

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

avec :

- X = (X1, X2, X3 ... Xn) — vecteur des variables d'entrée ;
- Y — variable de sortie ;
- a = (a1, a2, a3 ... an) — vecteur des valeurs des variables d'entrée ;
- d — valeur de la variable de sortie ;
- W — poids de la règle.

Méthodes de classe

Méthode de classe	Description
AggregationMethod	Définit le type d'aggrégation des conditions
Calculate	Calcule une inférence floue pour le système
DefuzzificationMethod	Définit le type de la méthode de défloutage
EmptyRule	Crée une règle floue de Mamdani vide basée sur le système actuel
ImplicationMethod	Définit le type de l'opérateur d'implication du système
Output	Retourne la liste des variables floues de Mamdani.
OutputByName	Retourne la variable floue de sortie de Mamdani ayant le nom spécifié.
ParseRule	Crée une règle floue de Mamdani basée sur une ligne spécifiée.
Rules	Retourne la liste des règles floues de Mamdani.

Méthodes héritées de la classe CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

AggregationMethod

Définit le type d'aggrégation des conditions.

```
void  AggregationMethod(
   AggregationMethod  value      // type de la méthode d'aggrégation
```

© 2000-2024, MetaQuotes Ltd.

```
   )
```

Paramètres

value

[in] Type de la méthode d'aggrégation des conditions.

Calculate

Calcule une inférence floue pour le système

```
CList*  Calculate(
   CList*  inputValues      // données d'entrée
   )
```

Paramètres

inputValues

[in] Données d'entrée pour les calculs.

Valeur de Retour

Résultat du calcul.

DefuzzificationMethod

Définit le type de la méthode de défloutage.

```
void  DefuzzificationMethod(
   DefuzzificationMethod  value      // type de la méthode de défloutage.
   )
```

Paramètres

value

[in] Type de la méthode de défloutage.

EmptyRule

Crée une règle floue de Mamdani vide basée sur le système actuel

```
CMamdaniFuzzyRule*  EmptyRule()
```

Valeur de Retour

Règle floue de Mamdani.

ImplicationMethod

Définit le type de l'opérateur d'implication des conditions.

```
void  ImplicationMethod(
   ImplicationMethod  value      // type de l'opérateur d'implication
```

© 2000-2024, MetaQuotes Ltd.

```
    )
```

Paramètres

value

[in] Type de l'opérateur d'implication des conditions.

Output

Retourne la liste des variables floues de Mamdani.

```
CList*  Output()
```

Valeur de Retour

Liste des variables floues.

OutputByName

Retourne la variable floue de sortie de Mamdani ayant le nom spécifié.

```
CFuzzyVariable*  OutputByName(
   const string  name      // nom de la variable floue
   )
```

Paramètres

name

[in] Nom de la variable floue.

Valeur de Retour

Variable floue de Mamdani ayant le nom spécifié.

ParseRule

Crée une règle floue de Mamdani basée sur une ligne spécifiée.

```
CMamdaniFuzzyRule*  ParseRule(
   const string  rule      // représentation textuelle d'une règle floue
   )
```

Paramètres

rule

[in] Représentation textuelle d'une régle floue de Mamdani.

Valeur de Retour

Règle floue de Mamdani.

Rules

© 2000-2024, MetaQuotes Ltd.

Retourne la liste des règles floues de Mamdani.

```
CList*  Rules()
```

Valeur de Retour

Liste des règles floues de Mamdani.

© 2000-2024, MetaQuotes Ltd.

Système de Sugeno

Le système de logique floue de Sugeno est l'un des deux types de base des systèmes flous. Les valeurs des variables de sortie sont définies comme une combinaison linéaire des variables d'entrée.

Description

Contrairement à la règle de Mamdani, la valeur d'une variable d'entrée est définie par une fonction linéaire à partir des entrées plutôt que par un terme flou. La règle de logique floue pour l'algorithme de Sugeno peut être décrit de la façon suivante :

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

avec :

- X = (X1, X2, X3 ... Xn) — vecteur des variables d'entrée ;
- Y — variable de sortie ;
- a = (a1, a2, a3 ... an) — vecteur des valeurs des variables d'entrée ;
- b = (b1, b2, b3 ... bn) — ratio de terme libre dans la fonction linéaire pourune valeur de sortie
- W — poids de la règle.

Méthodes de classe

Méthode de classe	Description
Calculate	Calcule une inférence floue pour le système
CreateSugenoFunction	Crée une fonction linéaire de Sugeno pour le système.
EmptyRule	Crée une règle floue de Sugeno vide basée sur le système actuel
Output	Retourne la liste des variables floues de Sugeno.
OutputByName	Retourne la variable floue de sortie de Sugeno ayant le nom spécifié.
ParseRule	Crée une règle floue de Sugeno basée sur une ligne spécifiée.
Rules	Retourne la liste des règles floues.

Méthodes héritées de la classe CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

Calculate

Calcule une inférence floue pour le système

```
CList*  Calculate(
   CList*&  inputValues       // données d'entrée
   )
```

© 2000-2024, MetaQuotes Ltd.

Paramètres

inputValues

[in] Données d'entrée pour les calculs.

Valeur de Retour

Résultat du calcul.

CreateSugenoFunction

Crée une fonction linéaire de Sugeno pour le système.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,          // nom de la fonction
   const double& coeffs[]       // ratios de la fonction
   )
```

Paramètres

name

[in] Nom de la fonction.

coeffs[]

[in] Ratios de la fonction.

Valeur de Retour

Fonction linéaire de Sugeno.

Note

La taille du tableau de ratios doit être égal au nombre d'entrées ou lui être supérieur de 1. Dans le premier cas, le terme libre de la fonction linéaire de Sugeno est égal à zéro, tandis que dans le deuxième cas, il est égal au dernier ratio.

CreateSugenoFunction

Crée une fonction linéaire de Sugeno pour le système.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,          // nom de la fonction
   CList*&       coeffs,        // liste de paires de variables floues - son ratio
   const double  constValue     // ratio du terme libre de la fonction
   )
```

Paramètres

name

[in] Nom de la fonction.

coeffs[]

[in] Ratios de la fonction.

© 2000-2024, MetaQuotes Ltd.

Valeur de Retour

Fonction linéaire de Sugeno.

EmptyRule

Crée une règle floue de Sugeno vide basée sur le système actuel

```
CSugenoFuzzyRule*  EmptyRule()
```

Valeur de Retour

Règle floue de Sugeno.

Output

Retourne la liste des variables floues de Sugeno.

```
CList*  Output()
```

Valeur de Retour

Liste des variables floues.

OutputByName

Retourne la variable floue de sortie de Sugeno ayant le nom spécifié.

```
CSugenoVariable*  OutputByName(
   const string  name      // nom de la variable floue
   )
```

Paramètres

name

Nom de la variable floue.

Valeur de Retour

Variable floue de Sugeno ayant le nom spécifié.

ParseRule

Crée une règle floue de Sugeno basée sur une ligne spécifiée.

```
CSugenoFuzzyRule*  ParseRule(
   const string  rule      // représentation textuelle d'une règle floue de Sugeno
   )
```

Paramètres

rule

[in] Représentation textuelle d'une régle floue de Sugeno.

© 2000-2024, MetaQuotes Ltd.

Valeur de Retour

Règle floue de Sugeno.

Rules

Retourne la liste des règles floues.

```
CList*  Rules()
```

Valeur de Retour

Liste des règles floues.

© 2000-2024, MetaQuotes Ltd.

Classe permettant de travailler avec les programmes OpenCL

La classe COpenCL est un wrapper facilitant le travail avec les fonctions OpenCL. Dans certains cas, l'utilisation de la GPU permet d'améliorer de façon substantielle la vitesse de calculs.

Des exemples d'utilisation de la classe pour des calculs basés sur des valeurs flottantes et doubles sont disponibles dans les répertoires correspondants du répertoire MQL5\Scripts\Examples\OpenCL\. Le code sources des programmes OpenCL sont situés dans les répertoires MQL5\Scripts\Examples\OpenCL\Double\Kernels et MQL5\Scripts\Examples\OpenCL\Float\Kernels.

- MatrixMult.mq5 - exemple de multiplication de matrices utilisant la mémoire globale et la mémoire locale
- BitonicSort.mq5 - exemple de tri en parallèle des éléments d'un tableau dans la GPU
- FFT.mq5 - exemple de calcul de la transformation de Fourier rapide
- Wavelet.mq5 - exemple de transformation en ondelettes de données utilisant l'ondelette de Morlet.

Il est recommandé d'écrire le code source OpenCL dans des fichiers CL séparés, pouvant être par la suite inclus dans le programme MQL5 en utilisant des variables de ressource.

Déclaration

```
class COpenCL
```

Titre

```
#include <OpenCL\OpenCL.mqh>
```

Méthode de la classe

Nom	Description
BufferCreate	Crée un buffer OpenCL à l'indice spécifié
BufferFree	Supprime le buffer à l'indice spécifié
BufferFromArray	Crée un buffer à l'indice spécifié à partir d'un tableau de valeurs
BufferRead	Lit un buffer OpenCL à l'indice spécifié dans un tableau
BufferWrite	Ecrit un tableau de valeurs dans le buffer à l'indice spécifié
Execute	Exécute le noyau OpenCL à l'indice spécifié
GetContext	Retourne le handle du contexte OpenCL
GetKernel	Retourne le handle de l'objet noyau à l'indice spécifié
GetKernelName	Retourne le nom de l'objet noyau à l'indice spécifié
GetProgram	Retourne le handle du programme OpenCL
Initialize	Initialise le programme OpenCL
KernelCreate	Crée un point d'entrée dans le programme OpenCL à l'indice spécifié
KernelFree	Supprime une fonction de démarrage OpenCL à l'indice spécifié

© 2000-2024, MetaQuotes Ltd.

Nom	Description
SetArgument	Définit un paramètre pour la fonction OpenCL à l'indice spécifié
SetArgumentBuffer	Définit un buffer OpenCL comme paramètre de la fonction OpenCL à l'indice spécifié
SetArgumentLocalMemory	Définit un paramètre dans la mémoire locale pour la fonction OpenCL à l'indice spécifié
SetBuffersCount	Définit le nombre de buffers
SetKernelsCount	Définit le nombre d'objets noyaux
Shutdown	Décharge le programme OpenCL
SupportDouble	Vérifie si les types de données à virgule flottante sont supportés sur le périphérique

© 2000-2024, MetaQuotes Ltd.

BufferCreate

Crée un buffer OpenCL à l'indice spécifié.

```
bool  BufferCreate(
   const int   buffer_index,      // indice du buffer
   const uint  size_in_bytes,     // taille du buffer en octets
   const uint  flags              //combinaison de flags définissant les propriétés du
   );
```

Paramètres

buffer_index

[in] Indice du buffer.

size_in_bytes

[in] Taille du buffer en octets.

flags

[in] Propriétés du buffer définies avec une combinaison de flags.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

BufferFree

Supprime le buffer à l'indice spécifié.

```
bool  BufferFree(
   const int  buffer_index      // indice du buffer
   );
```

Paramètres

buffer_index

[in] Indice du buffer.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

BufferFromArray

Crée un buffer à l'indice spécifié à partir d'un tableau de valeurs.

```
template<typename T>
bool  BufferFromArray(
   const int   buffer_index,          // indice du buffer
   T           &data[],               // tableau de valeurs
   const uint  data_array_offset,     // décalage dans le tableau de valeurs, en octet
   const uint  data_array_count,      // nombre de valeurs du tableau à écrire
   const uint  flags                  //combinaison de flags définissant les propriété
   );
```

Paramètres

buffer_index

[in] Indice du buffer.

&data[]

[in] Un tableau de valeurs à écrire dans le buffer OpenCL.

data_array_offset

[in] Décalage dans le tableau de valeurs en octets, à partir duquel l'écriture des valeurs commence.

data_array_count

[in] Le nombre de valeurs à écrire.

flags

[in] Propriétés du buffer définies avec une combinaison de flags.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

BufferRead

Lit un buffer OpenCL à l'indice spécifié dans un tableau.

```
template<typename T>
bool  BufferRead(
   const int   buffer_index,          // indice du buffer
   T           &data[],               // tableau de valeurs
   const uint  cl_buffer_offset,      // décalage dans le buffer OpenCL, en octets
   const uint  data_array_offset,     // décalage dans le tableau d'éléments
   const uint  data_array_count       // nombre de valeurs à lire depuis le buffer
   );
```

Paramètres

buffer_index

[in] Indice du buffer.

&data[]

[in] Tableau pour récupérer les valeurs du buffer OpenCL.

cl_buffer_offset

[in] Décalage dans le buffer OpenCL en octets, à partir duquel la lecture des valeurs commence.

data_array_offset

[in] Indice du premier élément du tableau pour écrire les valeurs du buffer OpenCL.

data_array_count

[in] Le nombre de valeurs à lire.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

BufferWrite

Ecrit un tableau de valeurs dans le buffer à l'indice spécifié.

```
template<typename T>
bool  BufferWrite(
   const int   buffer_index,          // indice du buffer
   T           &data[],               // tableau de valeurs
   const uint  cl_buffer_offset,      // décalage dans le buffer OpenCL, en octets
   const uint  data_array_offset,     // décalage dans le tableau d'éléments
   const uint  data_array_count       // nombre de valeurs du tableau à écrire
   );
```

Paramètres

buffer_index

[in] Indice du buffer.

&data[]

[in] Un tableau de valeurs à écrire dans le buffer OpenCL.

cl_buffer_offset

[in] Décalage dans le buffer OpenCL en octets, à partir duquel l'écriture des valeurs commence.

data_array_offset

[in] Indice du premier élément du tableau à partir duquel les valeurs du tableau sont écrites dans le buffer OpenCL.

data_array_count

[in] Le nombre de valeurs à écrire.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

Execute

Exécute le programme OpenCL à l'indice spécifié.

```
bool  Execute(
   const int   kernel_index,           // indice du noyau
   const int   work_dim,               // dimension de l'espace de la tâche
   const uint  &work_offset[],         // décalage initial dans l'espace de la tâche
   const uint  &work_size[]            // nombre total de tâches
   );
```

Exécute le noyau OpenCL à l'indice spécifié et avec le nombre de tâches du groupe local.

```
bool  Execute(
   const int   kernel_index,           // indice du noyau
   const int   work_dim,               // dimension de l'espace de la tâche
   const uint  &work_offset[],         // décalage initial dans l'espace de la tâche
   const uint  &work_size[],           // nombre total de tâches
   const uint  &local_work_size[]      // nombre de tâches dans le groupe local
   );
```

Paramètres

kernel_index

[in] Indice de l'objet noyau.

work_dim

[in] Dimension de l'espace des tâches.

&work_offset[]

[in][out] Décalage initial de l'espace des tâches. Passé par référence.

&work_size[]

[in][out] La taille du sous-ensemble de tâches. Passé par référence.

&local_work_size[]

[in][out] La taille du sous-ensemble local de tâches dans le groupe. Passé par référence.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

GetContext

Retourne le handle du contexte OpenCL.

```
int  GetContext();
```

Valeur de Retour

Handle du contexte OpenCL.

© 2000-2024, MetaQuotes Ltd.

GetKernel

Retourne le handle de l'objet noyau à l'indice spécifié.

```
int  GetKernel(
   const int  kernel_index      // indice du noyau
   );
```

Paramètres

kernel_index

[in] Indice de l'objet noyau.

Valeur de Retour

Handle sur l'objet noyau.

© 2000-2024, MetaQuotes Ltd.

GetKernelName

Retourne le nom de l'objet noyau à l'indice spécifié.

```
string  GetKernelName(
   const int  kernel_index      // indice du noyau
   );
```

Paramètres

kernel_index

[in] Indice de l'objet noyau.

Valeur de Retour

Nome de l'objet noyau.

© 2000-2024, MetaQuotes Ltd.

GetProgram

Retourne le handle du programme OpenCL.

```
int  GetProgram();
```

Valeur de Retour

Handle du programme OpenCL.

© 2000-2024, MetaQuotes Ltd.

Initialize

Initialise le programme OpenCL.

```
bool  Initialize(
   const string  program,           // handle du programme OpenCL
   const bool    show_log=true      // conserve un journal
   );
```

Paramètres

program

[in] Handle du programme OpenCL.

show_log=true

[in] Permet l'affichage des messages.

Valeur de Retour

Retourne true si l'initialisation a réussie. Sinon retourne false.

© 2000-2024, MetaQuotes Ltd.

KernelCreate

Crée un point d'entrée dans le programme OpenCL à l'indice spécifié.

```
bool  KernelCreate(
   const int     kernel_index,      // indice du noyau
   const string  kernel_name        // nom du noyau
   );
```

Paramètres

kernel_index

[in] Indice de l'objet noyau.

kernel_name

[in] Nom de l'objet noyau.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

KernelFree

Supprime une fonction de démarrage OpenCL à l'indice spécifié.

```
bool  KernelFree(
   const int  kernel_index      // indice du noyau
   );
```

Paramètres

kernel_index

[in] Indice de l'objet noyau.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

SetArgument

Définit un paramètre pour la fonction OpenCL à l'indice spécifié.

```
template<typename T>
bool  SetArgument(
   const int  kernel_index,      // indice du noyau
   const int  arg_index,         // indice de l'argument de la fonction
   T          value              // code source
   );
```

Paramètres

kernel_index

[in] Indice de l'objet noyau.

arg_index

[in] Indice de l'argument de la fonction.

value

[in] La valeur de l'argument de la fonction.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

SetArgumentBuffer

Définit un buffer OpenCL comme paramètre de la fonction OpenCL à l'indice spécifié

```
bool  SetArgumentBuffer(
   const int  kernel_index,      // indice du noyau
   const int  arg_index,         // indice de l'argument de la fonction
   const int  buffer_index       // indice du buffer
   );
```

Paramètres

kernel_index

[in] Indice de l'objet noyau.

arg_index

[in] Indice de l'argument de la fonction.

buffer_index

[in] Indice du buffer.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

SetArgumentLocalMemory

Définit un paramètre dans la mémoire locale pour la fonction OpenCL à l'indice spécifié.

```
bool  SetArgumentLocalMemory(
   const int  kernel_index,          // indice du noyau
   const int  arg_index,             // indice de l'argument de la fonction
   const int  local_memory_size      // taille de la mémoire locale
   );
```

Paramètres

kernel_index

[in] Indice de l'objet noyau.

arg_index

[in] Indice de l'argument de la fonction.

local_memory_size

[in] Taille de la mémoire locale.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

SetBuffersCount

Définit le nombre de buffers.

```
bool  SetBuffersCount(
   const int  total_buffers      // nombre de buffers
   );
```

Paramètres

total_buffers

[in] Le nombre total de buffers.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

SetKernelsCount

Définit le nombre d'objets noyaux

```
bool  SetKernelsCount(
   const int  total_kernels      // nombre de noyaux
   );
```

Paramètres

total_kernels

[in] Le nombre total de noyaux.

Valeur de Retour

Retourne true en cas d'exécution réussie, sinon - false.

© 2000-2024, MetaQuotes Ltd.

Shutdown

Décharge le programme OpenCL

```
void  Shutdown();
```

Valeur de Retour

Aucune valeur de retour.

© 2000-2024, MetaQuotes Ltd.

SupportDouble

Vérifie si les types de données à virgule flottante sont supportés sur le périphérique.

```
bool  SupportDouble();
```

Valeur de Retour

Retourne true si le périphérique supporte les types de données à virgule flottante.

© 2000-2024, MetaQuotes Ltd.

Classe de Base CObject

La classe CObject est la classe de base de la Bibliothèque Standard MQL5.

Description

La classe CObject permet à tous ses descendants de faire partie d'une liste chaînée. Elle propose également différentes méthodes virtuelles à implémenter dans ses classes descendantes.

Déclaration

```
class CObject
```

Titre

```
#include <Object.mqh>
```

Hiérarchie d'héritage

CObject

Descendants directs

CAccountInfo, CArray, CChart, CChartObject, CCurve, CDealInfo, CDictionary_Obj_Double, CDictionary_Obj_Obj, CDictionary_String_Obj, CExpertBase, CFile, CHistoryOrderInfo, CList, COrderInfo, CPositionInfo, CString, CSymbolInfo, CTerminalInfo, CTrade, CTreeNode, CWnd, ICondition, IExpression, IMembershipFunction, INamedValue, IParsableRule

Méthodes de Classe

Attributs	
Prev	Retourne la valeur de l'élèment précédent
Prev	Définit la valeur de l'élément précédent
Next	Retourne la valeur de l'élèment suivant
Next	Définit la valeur de l'élément suivant
Méthodes de comparaison	
virtual Compare	Retourne le résultat de la comparaison avec un autre objet
Entrée/Sortie	
virtual Save	Ecrit l'objet dans un fichierf
virtual Load	Charge l'objet depuis un fichier
virtual Type	Retourne le type de l'objet

© 2000-2024, MetaQuotes Ltd.

Prev

Retourne un pointeur sur l'élément précédent de la liste.

```
CObject*  Prev()
```

Valeur de Retour

Pointeur sur l'élément précédent de la liste. Si l'élément courant est le premier élément de la liste, retourne NULL.

Exemple :

```
//--- exemple d'utilisation de CObject::Prev()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- définit le lien entre les éléments
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utilise l'objet précédent
   CObject *object=object_second.Prev();
   //--- supprime les objets
   delete object_first;
   delete object_second;
  }
```

© 2000-2024, MetaQuotes Ltd.

Prev

Définit le pointeur vers l'élément précédent de la liste.

```
void  Prev(
   CObject*  object      // Pointeur vers l'élément précédent de la liste
   )
```

Paramètres

object

[in] Nouvelle valeur du pointeur sur l'élément précédent de la liste.

Exemple :

```
//--- exemple d'utilisation de CObject::Prev(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- définit le lien entre les éléments
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utilise les objets
   //--- ...
   //--- supprime les objets
   delete object_first;
   delete object_second;
  }
```

© 2000-2024, MetaQuotes Ltd.

Next

Retourne un pointeur sur l'élément suivant de la liste.

```
CObject*  Next()
```

Valeur de Retour

Pointeur sur l'élément suivant de la liste. Si l'élément courant est le dernier élément de la liste, retourne NULL.

Exemple :

```
//--- exemple d'utilisation de CObject::Next()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- définit le lien entre les éléments
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utilise l'objet suivant de la liste
   CObject *object=object_first.Next();
   //--- supprime les objets
   delete object_first;
   delete object_second;
  }
```

© 2000-2024, MetaQuotes Ltd.

Next

Définit le pointeur sur l'élément suivant de la liste.

```
void  Next(
   CObject*  object      // Pointeur sur l'élément suivant de la liste
   )
```

Paramètres

object

[in] Nouvelle valeur du pointeur sur l'élément suivant de la liste.

Exemple :

```
//--- exemple d'utilisation de CObject::Next(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- définit le lien entre les éléments
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utilise les objets
   //--- ...
   //--- supprime les objets
   delete object_first;
   delete object_second;
  }
```

© 2000-2024, MetaQuotes Ltd.

Compare

Compare les données d'un élément de la liste avec les données d'un autre élément de la liste.

```
virtual int  Compare(
   const CObject*  node,      // Node avec lequel comparer l'objet
   const int       mode=0     // Mode de comparaison
   ) const
```

Paramètres

node

[in] Pointeur vers un élément de la liste à comparer

mode=0

[in] Mode de comparaison

Valeur de Retour

0 - si les 2 éléments de la liste sont identiques, -1 - si l'élément de la liste est inférieur à l'élément de comparaison (node), 1 - si l'élément de la liste est supérieur à l'élément de comparaison (node).

Note

La méthode Compare () de la classe CObject retourne toujours 0 et n'exécute aucune action. Si vous désirez comparer des données de classes dérivées, la méthode Compare (...) doit être implémentée. Le paramètre 'mode' devrait être utilisé dans le cas de comparaison où plusieurs variantes sont possibles.

Exemple :

```
//--- exemple d'utilisation de CObject::Compare(...)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- définit le lien entre les éléments
```

© 2000-2024, MetaQuotes Ltd.

```
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- compare les objets
   int result=object_first.Compare(object_second);
   //--- supprime les objets
   delete object_first;
   delete object_second;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde un élément et ses données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

La méthode Save (int) de la classe CObject retourne toujours vrai et n'exécute aucune action. Si vous désirez sauvegarder des données de classes dérivées depuis un fichier, la méthode Save (int) doit être implémentée.

Exemple :

```
//--- exemple d'utilisation de CObject::Save(int)
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- définit les données de l'objet
   //--- . . .
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
        }
      FileClose(file_handle);
    }
  delete object;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge un élément et ses données dans la liste depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

La méthode Load (int) de la classe CObject retourne toujours vrai et n'exécute aucune action. Si vous désirez charger des données de classes dérivées depuis un fichier, la méthode Load (int) doit être implémentée.

Exemple :

```
//--- exemple d'utilisation de CObject::Load(int)
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  //--- utilise l'objet
  //--- . . .
  delete object;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type de l'objet (pour CObject - 0).

Exemple :

```
//--- exemple d'utilisation de CObject::Type()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object=new CObject;
   //---
   object=new CObject;
   if(object ==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type de l'objet
   int type=object.Type();
   //--- supprime l'objet
   delete object;
  }
```

© 2000-2024, MetaQuotes Ltd.

Collections de Données

Cette section contient les détails techniques d'utilisation des différentes structures de données (tableaux, listes chaînées, etc.) et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Utiliser des classes de structures de données vous fera gagner du temps lors de la création et du stockage de vos données dans différents formats (incluant les structures de données composites).

La Bibliothèque Standard MQL5 (en terme de données) est située dans le répertoire de travail du terminal dans le répertoire Include\Arrays.

Tableaux de Données

L'utiisation de classes de tableaux de données dynamiques vous fera gagner du temps lors de la création et du stockage de différents formats (incluant les tableaux à plusieurs dimensions).

La Bibliothèque Standard MQL5 (en terme de tableaux de données) est située dans le répertoire de travail du terminal dans le répertoire Include\Arrays.

Classe	Description
CArray	Classe de base du tableau dynamique de données
CArrayChar	Tableau dynamique de variables de type char ou uchar
CArrayShort	Tableau dynamique de variables de type short ou ushort
CArrayInt	Tableau dynamique de variables de type int ou uint
CArrayLong	Tableau dynamique de variables de type long ou ulong
CArrayFloat	Tableau dynamique de variables de type float
CArrayDouble	Tableau dynamique de variables de type double
CArrayString	Tableau dynamique de variables de type string
CArrayObj	Tableau dynamique de pointeurs de CObject
CList	Fournit la possibilité de travailler avec une liste d'instance de CObject et de ses descendants
CTreeNode	Fournit la possibilité de travailler avec les noeuds de l'arbre binaire CTree
CTree	Fournit la possibilité de travailler avec l'arbre binaire d'instances de classe de CTreeNode et

© 2000-2024, MetaQuotes Ltd.

Classe	Description
	de ses descendants

© 2000-2024, MetaQuotes Ltd.

CArray

La classe CArray est la classe de base des tableaux dynamiques de variables.

Description

La classe CArray permet d'effectuer des opérations sur les tableaux dynamiques de variables : allocation mémoire, tri et chargement/sauvegarde.

Déclaration

```
class CArray : public CObject
```

Titre

```
#include <Arrays\Array.mqh>
```

Hiérarchie d'héritage

CObject

CArray

Descendants directs

CArrayChar, CArrayDouble, CArrayFloat, CArrayInt, CArrayLong, CArrayObj, CArrayShort, CArrayString

Méthodes de Classe

Attributs	
Step	Retourne la taille du pas d'incrément du tableau
Step	Définit la taille du pas d'incrément du tableau
Total	Retourne le nombre total d'éléments dans le tableau
Available	Retourne le nombre d'éléments libres dans le tableau sans besoin d'allocation mémoire supplémentaire
Max	Retourne la taille possible maximum du tableau sans réallocation de mémoire
IsSorted	Retourne le tableau trié selon l'option spécifiée
SortMode	Retourne le mode de tri du tableau
Méthodes de suppression	
Clear	Supprime tous les éléments du tableau sans désallocation de la mémoire
Méthodes de tri	
Sort	Trie un tableau selon l'option spécifiée

© 2000-2024, MetaQuotes Ltd.

Attributs	
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Step

Retourne la taille du pas d'incrément du tableau.

```
int  Step() const
```

Valeur de Retour

Taille d'incrément du tableau.

Exemple :

```
//--- exemple d'utilisation de CArray::Step()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le pas
   int step=array.Step();
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Step

Définit la taille du pas d'incrément du tableau.

```
bool  Step(
   int  step      // pas
   )
```

Paramètres

step

[in] La nouvelle valeur du pas d'incrément du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le pas est inférieur ou égal à zéro.

Exemple :

```
//--- exemple d'utilisation de CArray::Step(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- définit la taille du pas
   bool result=array.Step(1024);
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Total

Retourne le nombre total d'éléments dans le tableau.

```
int  Total() const;
```

Valeur de Retour

Nombre d'éléments du tableau.

Exemple :

```
//--- exemple d'utilisation de CArray::Total()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie le total
   int total=array.Total();
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Available

Retourne le nombre d'éléments libres dans le tableau sans besoin d'allocation mémoire supplémentaire

```
int  Available() const
```

Valeur de Retour

Nombre d'éléments libres dans le tableau sans besoin d'allocation mémoire supplémentaire

Exemple :

```
//--- exemple d'utilisation de CArray::Available()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie les éléments disponibles
   int available=array.Available();
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Max

Retourne la taille possible maximum du tableau sans réallocation de mémoire

```
int  Max() const
```

Valeur de Retour

La taille possible maximum du tableau sans réallocation de la mémoire.

Exemple :

```
//--- exemple d'utilisation de CArray::Max()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie la taille maximum
   int max=array.Max();
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

IsSorted

Retourne le tableau trié selon l'option spécifiée.

```
bool  IsSorted(
   int  mode=0       // Mode de tri
   ) const
```

Paramètres

mode=0

[in] Mode de tri.

Valeur de Retour

Flag de la liste triée. Vrai si le tableau est trié suivant le mode spécifié, faux sinon.

Note

Le mode de tri du tableau ne peut pas être changé directement. La méthode Sort() réinitialise toutes les méthodes pour ajouter/insérer, excepté InsertSort(...).

Exemple :

```
//--- exemple d'utilisation de CArray::IsSorted()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie si le tableau est trié
   if(array.IsSorted())
     {
      //--- utilisation de méthodes sur le tableau trié
      //--- ...
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SortMode

Retourne le mode de tri du tableau.

```
int  SortMode() const;
```

Valeur de Retour

Mode de tri.

Exemple :

```
//--- exemple d'utilisation de CArray::SortMode()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie le mode du tri
   int sort_mode=array.SortMode();
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments du tableau sans désallocation de la mémoire

```
void  Clear()
```

Valeur de Retour

Aucune.

Exemple :

```
//--- exemple d'utilisation de CArray::Clear()
#include <Arrays\Array.mqh>
//---
void OnStart()
 {
 CArray *array=new CArray;
 //---
 if(array==NULL)
 {
 printf("Object create error");
 return;
 }
 //--- utilisation du tableau
 //--- ...
 //--- nettoyage du tableau
 array.Clear();
 //--- supprime le tableau
 delete array;
 }
```

© 2000-2024, MetaQuotes Ltd.

Sort

Trie un tableau selon l'option spécifiée.

```
void  Sort(
   int  mode=0       // Mode de tri
   )
```

Paramètres

mode=0

[in] Mode de tri du tableau.

Valeur de Retour

Aucune.

Note

Le tri d'un tableau est toujours fait en mode ascendant (du plus petit au plus grand). Pour les tableaux de types primitifs (CArrayChar, CArrayShort, etc.), le mode paramètre n'est pas utilisé. Pour le tableau CArrayObj, les différentes méthodes de tri devraient être implémentées dans la fonction Sort (int) des classes dérivées.

Exemple :

```
//--- exemple d'utilisation de CArray::Sort(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- tri par mode 0
   array.Sort(0);
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArray::Save(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArray::Load(...)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayChar

La classe CArrayChar est la classe des tableaux dynamiques de variables de type char ou uchar.

Description

La classe CArrayChar permet de travailler avec les tableaux dynamiques de variables de type char ou uchar. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayChar : public CArray
```

Titre

```
#include <Arrays\ArrayChar.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayChar

Méthodes de Classe

Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau
Méthodes de modification	

© 2000-2024, MetaQuotes Ltd.

Contrôle de la mémoire	
Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Méthodes concernant les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Cherche un élément égal à celui donné dans un tableau trié
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.
virtual Type	Retourne l'identifiant du type du tableau

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Reserve(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

true - en cas de succès, false - en cas d'échec dû à une tentative de mettre une taille inférieure à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Resize(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Shutdown()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   char  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Add(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const char&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::AddArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayChar*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayChar - source des éléments à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::AddArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   char  element,      // Elément à insérer
   int   pos           // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Insert(char,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const char&  src[],      // Tableau à insérerS
   int           pos        // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::InsertArray(const char &[],int)
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayChar*  src,      // Pointeur vers le tableau à insérer
   int           pos      // Position
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayChar - source des éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::InsertArray(const CArrayChar*,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- supprime le tableau source
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const char&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::AssignArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayChar*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayChar - source des éléments à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::AssignArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayChar *src  =new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2024, MetaQuotes Ltd.

```
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int   pos,          // Position
   char  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Update(int,char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,'A'))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer dans le tableau

shift

[in] Valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Shift(int,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Delete(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,      // Position du permier élément
   int  to         // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément à supprimer dans le tableau.

to

[in] Position du dernier élément à supprimer dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::DeleteRange(int,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
char  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, CHAR_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

CHAR_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::At(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      char result=array.At(i);
      if(result==CHAR_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const char&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::CompareArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const CArrayChar*  src      // Pointeur vers le tableau à comparer
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de la classe CArrayChar - source des éléments à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::CompareArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   char  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::InsertSort(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort('A'))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Search(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search('A')!=-1) printf("Element found");
   else                      printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchGreat(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat('A')!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchLess(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess('A')!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié

```
int  SearchGreatOrEqual(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchGreatOrEqual(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual('A')!=-1) printf("Element found");
   else                                  printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchLessOrEqual(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual('A')!=-1) printf("Element found");
   else                                 printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchFirst(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst('A')!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchLast(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast('A')!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchLinear(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear('A')!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Save(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Load(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%c'",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (77 pour CArrayChar).

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Type()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayShort

La classe CArrayLong est la classe des tableaux dynamiques de variables de type short ou ushort.

Description

La classe CArrayShort permet de travailler avec les tableaux dynamiques de variables de type short ou ushort. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayShort : public CArray
```

Titre

```
#include <Arrays\ArrayShort.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayShort

Méthodes de Classe

Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	
Update	Met à jour l'élément situé à la position spécifiée

© 2000-2024, MetaQuotes Ltd.

Contrôle de la mémoire	
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.
virtual Type	Retourne l'identifiant du type du tableau

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Reserve(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

true - en cas de succès, false - en cas d'échec dû à une tentative de mettre une taille inférieure à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Resize(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Shutdown()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   short  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Add(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const short&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::AddArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayShort*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayShort contenant les éléments source à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::AddArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insère un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   short  element,      // Elément à insérer
   int    pos           // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Insert(short,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const short&  src[],      // Tableau source
   int            pos         // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::InsertArray(const short &[],int)
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayShort*  src,      // Pointeur vers la source
   int            pos       // Position
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayShort contenant les éléments source à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::InsertArray(const CArrayShort*,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- supprime le tableau source
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const short&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::AssignArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayShort*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayShort contenant les éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::AssignArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayShort *src  =new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2024, MetaQuotes Ltd.

```
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int    pos,          // Position
   short  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Update(int,short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,100))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Positions
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer dans le tableau

shift

[in] Valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Shift(int,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Delete(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,      // Position du permier élément
   int  to         // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément à supprimer dans le tableau.

to

[in] Position du dernier élément à supprimer dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::DeleteRange(int,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
short  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, SHORT_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

SHORT_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::At(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      short result=array.At(i);
      if(result==SHORT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const short&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::CompareArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const CArrayShort*  src      // Pointeur vers la source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de la classe CArrayShort contenant les éléments sources de comparaison.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::CompareArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   short  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::InsertSort(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(100))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Search(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search(100)!=-1) printf("Element found");
   else                      printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchGreat(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat(100)!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchLess(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess(100)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchGreatOrEqual(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(100)!=-1) printf("Element found");
   else                                  printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchLessOrEqual(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(100)!=-1) printf("Element found");
   else                                 printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchFirst(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst(100)!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchLast(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast(100)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchLinear(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear(100)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Save(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Load(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (82 pour CArrayShort).

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Type()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayInt

La classe CArrayInt est la classe des tableaux dynamiques de variables de type int ou uint.

Description

La classe CArrayInt permet de travailler avec les tableaux dynamiques de variables de type int ou uint. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayInt : public CArray
```

Titre

```
#include <Arrays\ArrayInt.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayInt

Descendants directs

CSpreadBuffer

Méthodes de Classe

Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insère un élément dans le tableau à la position spécifiée
InsertArray	Insère un tableau d'éléments à la position spécifiée
InsertArray	Insère un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau

© 2000-2024, MetaQuotes Ltd.

Contrôle de la mémoire	
Méthodes de mise à jour	
Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.
virtual Type	Retourne l'identifiant du type du tableau

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Reserve(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

true - en cas de succès, false - en cas d'échec dû à une tentative de mettre une taille inférieure à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Resize(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Shutdown()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   int  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Add(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const int&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::AddArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayInt*  src      // Pointeur vers le tableau source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayInt - source des éléments à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::AddArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   int  element,      // Elément à insérer
   int  pos           // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Insert(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const int&  src[],      // Tableau source
   int          pos         // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::InsertArray(const int &[],int)
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayInt*  src,      // Pointeur vers le tableau source
   int          pos       // Position
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayInt - source des éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::InsertArray(const CArrayInt*,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- supprime le tableau source
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const int&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::AssignArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayInt*  src      // Pointeur vers le tableau source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayInt - source des éléments à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::AssignArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayInt *src  =new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
  //--- les tableaux sont identiques
  //--- supprime le tableau source
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int  pos,          // Position
   int  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Update(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,10000))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer dans le tableau

shift

[in] Valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Shift(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Delete(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,      // Position du permier élément
   int  to         // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément à supprimer dans le tableau.

to

[in] Position du dernier élément à supprimer dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::DeleteRange(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
int  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, INT_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

INT_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::At(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      int result=array.At(i);
      if(result==INT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const int&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::CompareArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const CArrayInt*  src      // Pointeur vers le tableau source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de la classe CArrayInt - source des éléments à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::CompareArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   int  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::InsertSort(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(10000))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Search(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search(10000)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchGreat(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat(10000)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchLess(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess(10000)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   int  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchGreatOrEqual(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(10000)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchLessOrEqual(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(10000)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt:: SearchFirst(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst(10000)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchLast(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast(10000)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchLinear(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear(10000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Save(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Load(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (82 pour CArrayInt).

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Type()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayLong

La classe CArrayLong est la classe des tableaux dynamiques de variables de type long ou ulong.

Description

La classe CArrayLong permet de travailler avec les tableaux dynamiques de variables de type long ou ulong. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayLong : public CArray
```

Titre

```
#include <Arrays\ArrayLong.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayLong

Descendants directs

CRealVolumeBuffer, CTickVolumeBuffer, CTimeBuffer

Méthodes de Classe

Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insère un élément dans le tableau à la position spécifiée
InsertArray	Insère un tableau d'éléments à la position spécifiée
InsertArray	Insère un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau

© 2000-2024, MetaQuotes Ltd.

Contrôle de la mémoire	
Méthodes de mise à jour	
Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.
virtual Type	Retourne l'identifiant du type du tableau

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Reserve(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

true - en cas de succès, false - en cas d'échec dû à une tentative de mettre une taille inférieure à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Resize(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Shutdown()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   long  element      // Elément à insérer
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Add(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const long&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::AddArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayLong*  src      // Pointeur vers le tableau source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayLong - source des éléments à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::AddArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   long  element,      // Elément à insérer
   int   pos           // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Insert(long,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const long&  src[],      // Tableau source
   int           pos         // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::InsertArray(const long &[],int)
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayLong*  src,      // Pointeur vers la source
   int           pos       // Position
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayLong - source des éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::InsertArray(const CArrayLong*,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- supprime le tableau source
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const long&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::AssignArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayLong*  src      // Pointeur vers le tableau source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayLong - source des éléments à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::AssignArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayLong *src  =new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2024, MetaQuotes Ltd.

```
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int   pos,          // Position
   long  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Update(int,long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,1000000))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer dans le tableau

shift

[in] Valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Shift(int,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Delete(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,      // Position du permier élément
   int  to         // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément à supprimer dans le tableau.

to

[in] Position du dernier élément à supprimer dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::DeleteRange(int,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
long  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, LONG_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

LONG_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::At(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      long result=array.At(i);
      if(result==LONG_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  Erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const long&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::CompareArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArrayconst

Compare le tableau avec un autre tableau.

```
bool  CompareArrayconst(
   const CArrayLong*  src      // Pointeur vers le tableau source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de la classe CArrayLong - source des éléments à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::CompareArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   long  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::InsertSort(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(1000000))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Search

Recherche un élément du tableau égal au modèle donné dans un tableau trié.

```
int  Search(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Search(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search(1000000)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Cherche un élément supérieur au modèle donné dans un tableau trié.

```
int  SearchGreat(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchGreat(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat(1000000)!=-1) printf("Element found");
   else                               printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchLess(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess(1000000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchGreatOrEqual(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(1000000)!=-1) printf("Element found");
   else                                      printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchLessOrEqual(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(1000000)!=-1) printf("Element found");
   else                                     printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Cherche le premier élément égal au modèle dans un tableau trié.

```
int  SearchFirst(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchFirst(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst(1000000)!=-1) printf("Element found");
   else                               printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchLast(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast(1000000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Recherche un élément du tableau égal au modèle donné dans le tableau.

```
int  SearchLinear(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchLinear(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear(1000000)!=-1) printf("Element found");
   else                                printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Save(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Load(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %I64",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (84 pour CArrayLong).

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Type()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayFloat

La classe CArrayFloat est la classe des tableaux dynamiques de variables de type float.

Description

La classe CArrayFloat permet de travailler avec les tableaux dynamiques de variables de type float. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayFloat : public CArray
```

Titre

```
#include <Arrays\ArrayFloat.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayFloat

Méthodes de Classe

Attributs	
Delta	Définit la tolérance de comparaison
Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau

© 2000-2024, MetaQuotes Ltd.

Attributs	
Méthodes de mise à jour	
Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.
virtual Type	Retourne l'identifiant du type du tableau

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Delta

Définit la tolérance de comparaison.

```
void  Delta(
   float  delta      // Tolérance
   )
```

Paramètres

delta

[in] La nouvelle valeur de tolérance de comparaison.

Valeur de Retour

Aucune

Note

Validité de la comparaison lors de la recherche. Les valeurs sont considérées égales si leur différence est inférieure ou égale à la tolérance. La tolérance par défaut est 0.0.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Delta(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- définit la tolérance de comparaison
   array.Delta(0.001);
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Reserve(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

true - en cas de succès, false - en cas d'échec dû à une tentative de mettre une taille inférieure à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Resize(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Shutdown()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   float  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Add(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const float&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::AddArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayFloat*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayFloat - source des éléments à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::AddArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   float  element,      // Elément à insérer
   int    pos           // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Insert(float,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const float&  src[],      // Tableau source
   int            pos         // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::InsertArray(const float &[],int)
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayFloat*  src,      // Pointeur vers le tableau source
   int            pos       // Position
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayFloat - source des éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::InsertArray(const CArrayFloat*,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- supprime le tableau source
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const float&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::AssignArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayFloat*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayFloat - source des éléments à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::AssignArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayFloat *src  =new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2024, MetaQuotes Ltd.

```
    delete src;
    //--- utilise le tableau
    //--- . . .
    //--- supprime le tableau
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int    pos,          // Position
   float  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Update(int,float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,100.0))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer dans le tableau

shift

[in] Valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Shift(int,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Delete(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,      // Position du permier élément
   int  to         // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément à supprimer dans le tableau.

to

[in] Position du dernier élément à supprimer dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::DeleteRange(int,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
float  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, FLT_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

FLT_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::At(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      float result=array.At(i);
      if(result==FLT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  Erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const float&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::CompareArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArrayconst

Copie les éléments d'un autre tableau.

```
bool  AssignArrayconst(
   const CArrayFloat*  src      // Pointeur vers la source
   ) const
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayFloat - source des éléments à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::CompareArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   float  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::InsertSort(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(100.0))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Search(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search(100.0)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchGreat(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat:: SearchLess(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchGreatOrEqual(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchLessOrEqual(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchFirst(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchLast(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchLinear(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear(100.0)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Save(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Load(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (87 pour CArrayFloat).

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Type()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayDouble

La classe CArrayDouble est la classe des tableaux dynamiques de variables de type double.

Description

La classe CArrayDouble permet de travailler avec les tableaux dynamiques de variables de type double. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayDouble : public CArray
```

Titre

```
#include <Arrays\ArrayDouble.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayDouble

Descendants directs

CDoubleBuffer

Méthodes de Classe

Attributs	
Delta	Définit la tolérance de comparaison
Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insère un élément dans le tableau à la position spécifiée
InsertArray	Insère un tableau d'éléments à la position spécifiée
InsertArray	Insère un tableau d'éléments à la position spécifiée

© 2000-2024, MetaQuotes Ltd.

Attributs	
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	
Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Recherche du min/max	
Minimum	Retourne l'index du plus petit élément du tableau dans l'intervalle spécifié
Maximum	Retourne l'index du plus grand élément du tableau dans l'intervalle spécifié
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié

© 2000-2024, MetaQuotes Ltd.

Attributs	
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.
virtual Type	Retourne l'identifiant du type du tableau

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Delta

Définit la tolérance de comparaison.

```
void  Delta(
   double  delta      // Tolérance
   )
```

Paramètres

delta

[in] La nouvelle valeur de tolérance de comparaison.

Valeur de Retour

Aucune

Note

Validité de la comparaison lors de la recherche. Les valeurs sont considérées égales si leur différence est inférieure ou égale à la tolérance. La tolérance par défaut est 0.0.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Delta(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- définit la tolérance de comparaison
   array.Delta(0.001);
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Reserve(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

true - en cas de succès, false - en cas d'échec dû à une tentative de mettre une taille inférieure à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Resize(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Shutdown()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   double  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter au tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Add(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const double&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur le tableau des éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::AddArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayDouble*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayDouble contenant les éléments source à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::AddArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insère un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   double  element,      // Elément à insérer
   int     pos           // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Insert(double,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const double&  src[],      // Tableau source
   int            pos         // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::InsertArray(const double &[],int)
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayDouble*  src,      // Pointeur vers la source
   int            pos       // Position
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayDouble contenant les éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::InsertArray(const CArrayDouble*,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- supprime le tableau source
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const double&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur le tableau des éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::AssignArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayDouble*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayDouble contenant les éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::AssignArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayDouble *src  =new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2024, MetaQuotes Ltd.

```
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int     pos,          // Position
   double  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Update(int,double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,100.0))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer

shift

[in] La valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Shift(int,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Delete(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,      // Position du permier élément
   int  to         // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément du tableau à supprimer.

to

[in] Position du dernier élément du tableau à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::DeleteRange(int,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
double  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, DBL_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

DBL_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::At(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      double result=array.At(i);
      if(result==DBL_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  Erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const double&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::CompareArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const CArrayDouble*  src      // Pointeur vers la source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de la classe CArrayDouble contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::CompareArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Minimum

Retourne l'index du plus petit élément du tableau dans l'intervalle spécifié.

```
int  Minimum(
   int  start,      // index du début de l'intervalle
   int  count       // nombre d'éléments à traiter
   ) const
```

Paramètres

start

[in] Index de départ du tableau.

count

[in] Nombre d'éléments à traiter.

Valeur de retour

Index du plus petit élément du tableau dans l'intervalle spécifié.

© 2000-2024, MetaQuotes Ltd.

Maximum

Retourne l'index du plus grand élément du tableau dans l'intervalle spécifié.

```
int  Maximum(
   int  start,      // index du début de l'intervalle
   int  count       // nombre d'éléments à traiter
   ) const
```

Paramètres

start

[in] Index de départ du tableau.

count

[in] Nombre d'éléments à traiter.

Valeur de retour

Index du plus grand élément du tableau dans l'intervalle spécifié.

© 2000-2024, MetaQuotes Ltd.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   double  element      // Elément à insérer
   )
```

Paramètres

element

[in] valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::InsertSort(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(100.0))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Search(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search(100.0)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchGreat(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble:: SearchLess(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchGreatOrEqual(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchLessOrEqual(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchFirst(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchLast(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchLinear(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear(100.0)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Save(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Load(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (87 pour CArrayDouble).

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Type()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayString

La classe CArrayString est la classe des tableaux dynamiques de variables de type string.

Description

La classe CArrayString permet de travailler avec les tableaux dynamiques de variables de type string. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayString : public CArray
```

Titre

```
#include <Arrays\ArrayString.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayString

Méthodes de Classe

Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Définit une nouvelle taille du tableau (plus petite)
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	
Update	Met à jour l'élément situé à la position spécifiée

© 2000-2024, MetaQuotes Ltd.

Contrôle de la mémoire	
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Opérations sur les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Cherche un élément égal à celui donné dans un tableau trié
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal au modèle donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal au modèle donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle donné dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle donné dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.
virtual Type	Retourne l'identifiant du type du tableau

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Reserve(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

true - en cas de succès, false - en cas d'échec dû à une tentative de mettre une taille inférieure à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayString::Resize(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Shutdown()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   string  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter au tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Add(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(IntegerToString(i)))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const string&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur le tableau des éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::AddArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayString*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayString contenant les éléments source à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::AddArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   string  element,      // Elément à insérer
   int     pos           // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Insert(string,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(IntegerToString(i),0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const string&  src[],      // Tableau source
   int             pos        // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::InsertArray(const string &[],int)
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayString*  src,      // Pointeur vers la source
   int             pos      // Position
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayString contenant les éléments source à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::InsertArray(const CArrayString*,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- supprime le tableau source
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const string&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur le tableau des éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::AssignArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayString*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayString contenant les éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::AssignArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayString *src  =new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2024, MetaQuotes Ltd.

```
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int     pos,          // Position
   string  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Update(int, string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,"ABC"))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer

shift

[in] La valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Shift(int,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Delete(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,      // Position du permier élément
   int  to         // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément du tableau à supprimer.

to

[in] Position du dernier élément du tableau à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::DeleteRange(int,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
string  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, " (chaîne vide) si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

" (chaîne vide) peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayString::At(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      string result=array.At(i);
      if(result=="" && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- Erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const string&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayString::CompareArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArrays(
   const CArrayString*  src      // Pointeur vers la source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayString contenant les éléments source de comparaison.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayString::CompareArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   string  element      // Elément à insérer
   )
```

Paramètres

element

[in] valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayString::InsertSort(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort("ABC"))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Search(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search("ABC")!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString::SearchGreat(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat("ABC")!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString:: SearchLess(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess("ABC")!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString:: SearchGreatOrEqual(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual("ABC")!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString:: SearchLessOrEqual(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual("ABC")!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString:: SearchFirst(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst("ABC")!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString:: SearchLast(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast("ABC")!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::SearchLinear(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear("ABC")!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Save(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(IntegerToString(i));
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Load(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%s'",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (89 pour CArrayString).

Exemple :

```
//--- exemple d'utilisation de CArrayString::Type()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayObj

La classe CArrayObj est la classe des tableaux dynamiques de pointeurs vers des instances de CObject et de ses descendants.

Description

La classe CArrayObj permet de travailler avec un tableau dynamique de pointeurs vers des instances de CObject et de ses descendants. Ceci permet de travailler comme avec des tableaux de types de données primitifs à plusieurs dimensions.

La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Il y a quelques subtilités de la class CArrayObj.

Déclaration

```
class CArrayObj : public CArray
```

Titre

```
#include <Arrays\ArrayObj.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj

Descendants directs

CIndicators, CSeries

Méthode de Classe

Attributs	
FreeMode	Retourne le flag de gestion de la mémoire
FreeMode	Définit le flag de gestion de la mémoire
Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire.
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau

© 2000-2024, MetaQuotes Ltd.

Attributs	
AddArray	Ajoute un élément à la fin du tableau
Insert	Insère un élément dans le tableau à la position spécifiée
InsertArray	Insère un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	
Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Detach	Retourne l'élément du tableau situé à la position spécifiée et l'enlève du tableau
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Clear	Supprime tous les éléments du tableau sans désallouer la mémoire
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
Opérations avec les tableaux triés	
InsertSort	Insère un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié

© 2000-2024, MetaQuotes Ltd.

Attributs	
Entrée/Sortie	
Save	Sauvegarde le tableau de données dans un fichier
Load	Charge les données du tableau depuis un fichier.
Type	Retourne l'identifiant du type du tableau

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Une application pratique des tableaux sont les descendants de la classe CObject (incluant toutes les classes de la bibliothèque standard).

Par exemple, considérons un tableau à 2 dimensions :

```
#include <Arrays\ArrayDouble.mqh>
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int i,j;
   int first_size=10;
   int second_size=100;
//--- création du tableau
   CArrayObj     *array=new CArrayObj;
   CArrayDouble *sub_array;
//---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
//--- création des sous-tableaux
   for(i=0;i<first_size;i++)
     {
      sub_array=new CArrayDouble;
      if(sub_array==NULL)
        {
         delete array;
         printf("Object create error");
         return;
        }
      //--- remplissage du tableau
      for(j=0;j<second_size;j++)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         sub_array.Add(i*j);
        }
      array.Add(sub_array);
     }
//--- tableau créé avec succès
   for(i=0;i<first_size;i++)
     {
      sub_array=array.At(i);
      for(j=0;j<second_size;j++)
        {
         double element=sub_array.At(j);
         //--- utilisation de l'élément du tableau
        }
     }
   delete array;
  }
```

Subtilités

La classe possède un mécanisme de contrôle dynamique de la mémoire, veillez donc à faire attention lorsque vous travaillez avec des éléments dans un tableau.

Le mécanisme de gestion de la mémoire peut être activé ou désactivé en utilisant la méthode FreeMode (bool). Par défaut, le mécanisme est activé.

De la même façon, il y a deux options de gestion de la classe CArrayObj :

1. Mécanisme de gestion de la mémoire activé. (par défaut)

Dans ce cas, la classe CArrayObj prend la responsabilité de libérer les éléments de la mémoire après leur suppression du tableau. Dans ce mode, l'utilisateur ne devrait pas libérer les éléments d'un tableau.

Exemple :

```
   int i;
//--- Création d'un tableau
   CArrayObj *array=new CArrayObj;
//--- Remplissage des éléments du tableau
   for(i=0;i<10;i++) array.Add(new CObject);
//--- Fait quelque chose
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- Action sur un élément
      . . .
     }
//--- Supprime le tableau et ses éléments
   delete array;
```

© 2000-2024, MetaQuotes Ltd.

2. Mécanisme de gestion de la mémoire désactivé.

Dans ce cas, la classe CArrayObj ne prend pas la responsabilité de libérer les éléments de la mémoire après leur suppression du tableau. Dans ce mode, l'utilisateur doit libérer les éléments d'un tableau.

Exemple :

```
   int i;
//--- Création d'un tableau
   CArrayObj *array=new CArrayObj;
//--- Désactive le mécanisme de la gestion de la mémoire
   array.FreeMode(false);
//--- Remplissage des éléments du tableau
   for(i=0;i<10;i++) array.Add(new CObject);
//--- Fait quelque chose
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- Action sur un élément
      . . .
     }
//--- Supprime les éléments du tableau
   while(array.Total()) delete array.Detach();
//--- Supprime le tableau vide
   delete array;
```

© 2000-2024, MetaQuotes Ltd.

FreeMode

Retourne le mode de gestion de la mémoire.

```
bool  FreeMode() const
```

Valeur de Retour

Mode de gestion de la mémoire.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::FreeMode()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le mode de gestion de la mémoire
   bool array_free_mode=array.FreeMode();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

FreeMode

Définit le mode de gestion de la mémoire.

```
void  FreeMode(
   bool  mode       // Nouveau mode
   )
```

Paramètres

mode

[in] Nouveau mode de gestion de la mémoire.

Valeur de Retour

Aucune.

Note

La définition du mode de gestion de la mémoire est un point important de l'utilisation de la classe CArrayObj. Puisque les éléments du tableau sont des pointeurs vers des objets, il est important de savoir comment les gérer lorsqu'ils sont enlevés du tableau.

Si le mode est activé, enlever un élément du tableau déclenche sa suppression par un appel à l'opérateur delete. si le mode n'est pas activé, le pointeur vers l'objet existe toujours quelque part dans le programme et doit être libéré par le programme.

Si le mode de gestion de la mémoire est réinitialisé, le programmeur doit prendre la responsabilité de la suppression des éléments de la liste avant la fin du programme pour libérer la mémoire utilisée lors de la création de l'élément avec l'opérateur new. Avec de gros volumes de données, ceci peut empêcher votre terminal de fonctionner.

Avec de gros volumes de données, cela peut amener à des problèmes d'utilisation du terminal. Il y a également un autre risque.

Avec les listes de pointeurs, les éléments sont stockés sous forme de variables locales. Après la suppression de la liste, ne pas supprimer ces variables peut provoquer des erreurs critiques et des crashs du programme. Par défaut, le mode de gestion de la mémoire est activé, la classe de la liste est responsable de la libération de la mémoire de ses éléments.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::FreeMode(bool)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- réinitialise le flag du mode de suppression
   array.FreeMode(false);
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Reserve(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

true - en cas de succès, false - en cas d'échec dû à une tentative de mettre une taille inférieure à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments en excés situés sur la droite sont perdus. La mémoire des éléments perdus est libérée ou pas suivant le mode de gestion de la mémoire.

Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé avec le pas définit avec la méthode Step (int), ou sinon avec un pas de 16 (par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Resize(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments du tableau sans désallouer la mémoire

```
void  Clear()
```

Valeur de Retour

Aucune.

Note

Si le mode de gestion de la mémoire est activé, les éléments supprimés sont exemptés.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Clear()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- vide le tableau
   array.Clear();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Note

Si le mode de gestion de la mémoire est activé, les éléments supprimés sont exemptés.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Shutdown()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CreateElement

Crée un nouvel élément du tableau à la position spécifiée.

```
bool  CreateElement(
   int  index      // Position
   )
```

Paramètres

index

[in] Position d'insertion de l'élément dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être créé.

Note

La méthode CreateElement (int) de la classe CArrayObj retourne toujours faux et n'exécute aucune action. La méthode CreateElement (int) devrait être implémenté si nécessaire dans les classes dérivées.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::CreateElement(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int        size=100;
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- remplissage du tableau
   array.Reserve(size);
   for(int i=0;i<size;i++)
     {
      if(!array.CreateElement(i))
        {
         printf("Element create error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
```

© 2000-2024, MetaQuotes Ltd.

```
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   CObject*  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter au tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Note

L'élément n'est pas ajouté au tableau si le pointeur est invalide (NULL par exemple).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Add(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(new CObject))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayObj *  src      // Pointeur vers le tableau source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayDouble contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Note

L'ajout des éléments dans le tableau est en fait une copie de pointeurs. En appelant la méthode, plusieurs variables peuvent donc contenir un pointeur vers un objet dynamique.

```
//--- exemple
extern bool       make_error;
extern int        error;
extern CArrayObj *src;
//--- Crée une nouvelle instance de CArrayObj
//--- La gestion de la mémoire est par défaut activée
CArrayObj *array=new CArrayObj;
//--- Ajoute (copie) les éléments du tableau source
if(array!=NULL)
   bool result=array.AddArray(src);
if(make_error)
  {
   //--- Effectue une action déclenchant une erreur
   switch(error)
     {
      case 0:
         //--- Enlève le tableau source sans vérifier son mode de gestion de la mémoi
         delete src;
         //--- Résultat :
         //--- Il est possible de référencer un élément par un pointeur invalide dans
         break;
      case 1:
         //--- Désactive le mode de gestion de la mémoire dans le tableau source
         if(src.FreeMode()) src.FreeMode(false);
         //--- Mais ne supprime pas le tableau source
         //--- Résultat :
         //--- Après avoir supprimé le tableau destination, il est possible de référer
         break;
      case 2:
```

© 2000-2024, MetaQuotes Ltd.

```
          //--- Désactive le mode de gestion de la mémoire dans le tableau source
          src.FreeMode(false);
          //--- Désactive le mode de gestion de la mémoire dans le tableau destination
          array.FreeMode(false);
          //--- Résultat :
          //--- Après la fin du programme, une "fuite mémoire" apparaît
          break;
      }
   }
else
   {
    //--- Désactive le mode de gestion de la mémoire dans le tableau source
    if(src.FreeMode()) src.FreeMode(false);
    //--- Supprime le tableau source
    delete src;
    //--- Résultat :
    //--- Adresser un élément du tableau source sera correct
    //--- La suppression du tableau destination supprimera tous ses éléments
   }
```

Exemple :

```
//--- exemple d'utilisation de CArrayObj::AddArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
   {
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
      {
       printf("Object create error");
       return;
      }
    //--- crée le tableau source
    CArrayObj *src=new CArrayObj;
    if(src==NULL)
      {
       printf("Object create error");
       delete array;
       return;
      }
    //--- réinitialise le flag du mode de suppression
    src.FreeMode(false);
    //--- remplit le tableau source
    //--- . . .
    //--- ajoute un autre tableau
    if(!array.AddArray(src))
```

© 2000-2024, MetaQuotes Ltd.

```
    {
     printf("Array addition error");
     delete src;
     delete array;
     return;
    }
  //--- supprime le tableau source sans les éléments
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   CObject*  element,      // Elément à insérer
   int       pos           // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Note

L'élément n'est pas ajouté au tableau si le pointeur est invalide (NULL par exemple).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Insert(CObject*,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(new CObject,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
```

© 2000-2024, MetaQuotes Ltd.

```
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const CArrayObj*  src,      // Pointeur vers la source
   int               pos       // Position
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayObj contenant les éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Note

Voir également : CArrayObj::AddArray(const CArrayObj*).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::InsertArray(const CArrayObj*,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- réinitialise le flag du mode de suppression
   src.FreeMode(false);
   //--- remplit le tableau source
   //--- . . .
   //--- insére un autre tableau
```

© 2000-2024, MetaQuotes Ltd.

```
  if(!array.InsertArray(src,0))
    {
     printf("Array inserting error");
     delete src;
     delete array;
     return;
    }
  //--- supprime le tableau source sans les éléments
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayObj*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayObj contenant les éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Note

Si le tableau destination de la fonction AssignArray n'est pas vide, tous ses éléments seront supprimés du tableau, et si le mode de gestion de la mémoire est activé, la mémoire sera libérée. Le tableau destination est la copie exacte du tableau source. Voir éfalement CArrayObj::AddArray(const CArrayObj*).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::AssignArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- réinitialise le flag du mode de suppression
   src.FreeMode(false);
   //--- remplit le tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
```

© 2000-2024, MetaQuotes Ltd.

```
    {
     printf("Array assigned error");
     delete src;
     delete array;
     return;
    }
  //--- les tableaux sont identiques
  //--- supprime le tableau source sans les éléments
  delete src;
  //--- utilise le tableau
  //--- . . .
  //--- supprime le tableau
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int       pos,          // Position
   CObject*  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Note

L'élément n'est pas modifié si le pointeur passé en paramètre est invalide (NULL par exemple). Si la mode de gestion de la mémoire est activé, la mémoire est libérée si besoin.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Update(int,CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,new CObject))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer

shift

[in] La valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Shift(int,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Detach

Détache l'élément du tableau situé à la position spécifiée.

```
CObject*  Detach(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

Pointeur sur l'objet détaché en cas de succès, NULL si l'élément ne peut pas être détaché.

Note

Lorsqu'un élément du tableau est détaché, il n'est pas supprimé, quelque soit le mode de gestion de la mémoire. Le pointeur vers l'élément du tableau est retiré de la liste des éléments à libérer.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Detach(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   CObject *object=array.Detach(0);
   if(object==NULL)
     {
      printf("Detach error");
      delete array;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- supprime l'élément
   delete object;
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Note

Si le mode de gestion de la mémoire est activé, les éléments supprimés sont exemptés.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Delete(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,      // Position du permier élément
   int  to         // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément du tableau à supprimer.

to

[in] Position du dernier élément du tableau à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Note

Si le mode de gestion de la mémoire est activé, les éléments supprimés sont exemptés.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::DeleteRange(int,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
CObject*  At(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, NULL si la position donnée n'existe pas.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::At(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      CObject *result=array.At(i);
      if(result==NULL)
        {
         //--- Erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const CArrayObj*  src      // Pointeur vers la source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayObj contenant les éléments source de comparaison.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::CompareArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- remplit le tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   CObject*  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Note

L'élément n'est pas ajouté au tableau si le pointeur est invalide (NULL par exemple).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::InsertSort(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(new CObject))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Search(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.Search(sample)!=-1) printf("Element found");
   else                         printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::SearchGreat(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchGreat(sample)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj:: SearchLess(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLess(sample)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::SearchGreatOrEqual(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(sample)!=-1) printf("Element found");
   else                                     printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj:: SearchLessOrEqual(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(sample)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::SearchFirst(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchFirst(sample)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj:: SearchLast(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLast(sample)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Save(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

Lors de la lecture du fichier, chaque élément du tableau est créé en appelant la méthode CArrayObj::CreateElement(int).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Load(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- utilisation des éléments du tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (7778 pour CArrayObj).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Type()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CList

La classe CList est une liste d'instance de classes CObject et de ses descendants.

Description

La classe CList fournit la possibilité de travailler avec une liste d'instance de CObject et de ses descendants La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans la liste, de tri et de recherche dans la liste triée. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Il y a quelques subtilités à connaître lorsque l'on travaille avec la classe CList. La classe possède un mécanisme de contrôle dynamique de la mémoire, veillez donc à faire attention lorsque vous travaillez avec des éléments dans la liste.

Les subtilités du mécanisme de gestion de la mémoire sont similaires à celles décrites pour la classe CArrayObj.

Déclaration

```
class CList : public CObject
```

Titre

```
#include <Arrays\List.mqh>
```

Hiérarchie d'héritage

CObject
  CList

Méthodes de Classe

Attributs	
FreeMode	Retourne le mode de gestion de la mémoire lors de la suppression d'éléments de la liste.
FreeMode	Retourne le mode de gestion de la mémoire lors de la suppression d'éléments de la liste.
Total	Retourne le nombre d'éléments dans la liste
IsSorted	Retourne le flag permettant de savoir si la liste est triée ou pas
SortMode	Retourne le mode de tri
Méthodes de création	
CreateElement	Crée un nouvel élément dans la liste
Méthode d'ajout	
Add	Ajoute un élément à la fin de la liste

© 2000-2024, MetaQuotes Ltd.

Attributs	
Insert	Insére un élément dans la liste à la position spécifiée
Méthodes de suppression	
DetachCurrent	Détache un élément de la liste situé à la position spécifiée sans le supprimer "physiquement"
DeleteCurrent	Supprime l'élément courant de la liste
Delete	Supprime l'élément de la liste situé à la position spécifiée
Clear	Supprime tous les éléments de la liste
Navigation	
IndexOf	Retourne la position de l'élément dans la liste
GetNodeAtIndex	Retourne l'élément de la liste correspondant à la position spécifiée
GetFirstNode	Retourne le premier élément de la liste
GetPrevNode	Retourne l'élément précédent dans la liste
GetCurrentNode	Retourne l'élément courant de la liste
GetNextNode	Retourne le prochain élément de la liste
GetLastNode	Retourne le dernier élément
Méthodes de tri	
Sort	Trie la liste
MoveToIndex	Déplace l'élément courant de la liste à la position spécifiée
Exchange	Intervertit deux éléments de la liste
Méthodes de comparaison	
CompareList	Compare la liste avec une autre liste
Méthodes de recherche	
Search	Cherche un élément égal au modèle donné dans une liste triée
Entrée/Sortie	
virtual Save	Sauvegarde les données de la liste dans un fichier.
virtual Load	Charge les données de la liste depuis un fichier
virtual Type	Retourne l'identifiant du type de la liste.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

FreeMode

Retourne le mode de gestion de la mémoire lors de la suppression d'éléments de la liste.

```
bool  FreeMode() const
```

Valeur de Retour

Mode de gestion de la mémoire.

Exemple :

```
//--- exemple d'utilisation de CList::FreeMode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le mode de gestion de la mémoire
   bool list_free_mode=list.FreeMode();
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

FreeMode

Définit le mode de gestion de la mémoire lors de la suppression d'éléments de la listel.

```
void  FreeMode(
   bool  mode      // Nouvelle valeur
   )
```

Paramètres

mode

[in] Nouveau mode de gestion de la mémoire.

Note

La gestion de la mémoire est un élément important de la classe CList. Puisque les éléments de la liste sont des pointerus vers des objets, il est important de savoir quoi en faire lorsqu'il est retiré de la liste. Si le mode de gestion de la mémoire est activé, l'élément enlevé de la liste est alors automatiquement supprimé par un appel à l'opérateur delete. Si le mode de gestion de la mémoire n'est pas activé, le pointeur existe toujours lorsque l'élément est enlevé de la liste et c'est au programme de se charger de le supprimer et de libérer la mémoire.

Si le mode de gestion de la mémoire est réinitialisé, le programmeur doit prendre la responsabilité de la suppression des éléments de la liste avant la fin du programme pour libérer la mémoire utilisée lors de la création de l'élément avec l'opérateur new. Avec de gros volumes de données, ceci peut empêcher votre terminal de fonctionner.

Si le mode de gestion de la mémoire n'est pas réinitialisé, un autre risque subsiste. Avec les listes de pointeurs, les éléments sont stockés sous forme de variables locales. Après la suppression de la liste, ne pas supprimer ces variables peut provoquer des erreurs critiques et des crashs du programme. Par défaut, le mode de gestion de la mémoire est activé, la classe de la liste est responsable de la libération de la mémoire de ses éléments.

Exemple :

```
//--- exemple d'utilisation de CList::FreeMode(bool)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réinitialise le flag du mode de suppression
   list.FreeMode(false);
   //--- utilisation de la liste
   //--- . . .
   //--- supprime la liste
```

© 2000-2024, MetaQuotes Ltd.

```
    delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Total

Retourne le nombre d'éléments dans la liste.

```
int  Total() const
```

Valeur de Retour

Nombre d'éléments dans la liste.

Exemple :

```
//--- exemple d'utilisation de CList::Total()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie le nombre total
   int total=list.Total();
   //--- utilisation de la liste
   //--- ...
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

IsSorted

Retourne le flag permettant de savoir si la liste est triée ou pas

```
bool  IsSorted(
   int  mode=0       // Mode de tri
   ) const
```

Paramètres

mode=0

[in] Mode de tri à vérifier

Valeur de Retour

Flag de la liste triée. Si la liste est triée suivant le mode de tri spécifié? vrai, sinon? faux.

Note

Le mode de tri d'une liste ne peut pas être changé directement. Le mode est défini avec la méthode Sort (int) et réinitialise toutes les méthodes d'ajout / insertion.

Exemple :

```
//--- exemple d'utilisation de CList::IsSorted()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie le mode de tri
   if(list.IsSorted(0))
     {
      //--- utilisation de méthodes sur les listes triées
      //--- ...
     }
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

SortMode

Retourne le mode de tri.

```
int  SortMode() const
```

Valeur de Retour

Mode de tri, ou -1 si la liste n'est pas triée.

Exemple :

```
//--- exemple d'utilisation de CList::SortMode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie le mode de tri
   int sort_mode=list.SortMode();
   //--- utilisation de la liste
   //--- ...
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

CreateElement

Crée un nouvel élément dans la liste.

```
CObject*  CreateElement()
```

Valeur de Retour

Pointeur sur le nouvel élément créé, NULL si l'élément ne peut pas être créé.

Note

La méthode CreateElement (int) de la classe CList retourne toujours NULL et n'exécute aucune action. La méthode CreateElement () devrait être implémenté si nécessaire dans les classes dérivées.

Exemple :

```
//--- exemple d'utilisation de CList::CreateElement(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    size=100;
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- remplissage de la liste
   for(int i=0;i<size;i++)
     {
      CObject *object=list.CreateElement();
      if(object==NULL)
        {
         printf("Element create error");
         delete list;
         return;
        }
      list.Add(object);
     }
   //--- utilisation de la liste
   //--- . . .
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à la fin de la liste.

```
int  Add(
   CObject*  element      // Elément à ajouter
   )
```

Paramètres

element

[in] Valeur de l'élément à ajouter à la liste.

Valeur de Retour

La position de l'élément ajouté en cas de succès, -1 en cas d'erreur.

Note

L'élément n'est pas ajouté à la liste si le pointeur passé en paramètre est invalide (NULL par exemple).

Exemple :

```
//--- exemple d'utilisation de CList::Add(Cobject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoue 100 éléments
   for(int i=0;i<100;i++)
     {
      if(list.Add(new CObject)==-1)
        {
         printf("Element addition error");
         delete list;
         return;
        }
     }
   //--- utilisation de la liste
   //--- . . .
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insére un élément dans la liste à la position spécifiée.

```
int  Insert(
   CObject*  element,      // Elément à insérer
   int       pos           // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans la liste.

pos

[in] Position d'insertion de l'élément dans la liste

Valeur de Retour

La position de l'élément inséré en cas de succès, -1 en cas d'erreur.

Note

L'élément n'est pas ajouté à la liste si le pointeur passé en paramètre est invalide (NULL par exemple).

Exemple :

```
//--- exemple d'utilisation de CList::Insert(CObject*,int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére 100 éléments
   for(int i=0;i<100;i++)
     {
      if(list.Insert(new CObject,0)==-1)
        {
         printf("Element insert error");
         delete list;
         return;
        }
     }
   //--- utilisation de la liste
   //--- . . .
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

DetachCurrent

Détache l'élément de la liste situé à la position spécifiée sans le supprimer "physiquement".

```
CObject*  DetachCurrent()
```

Valeur de Retour

Pointeur sur l'objet détaché en cas de succès, NULL si l'élément ne peut pas être détaché.

Note

Lorsqu'un élément de la liste est détaché, il n'est pas supprimé, quelque soit le mode de gestion de la mémoire. Le pointeur vers l'élément de la liste est retiré de la liste des éléments à libérer.

Exemple :

```
//--- exemple d'utilisation de CList::DetachCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.DetachCurrent();
   if(object==NULL)
     {
      printf("Detach error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- supprime l'élément
   delete object;
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteCurrent

Supprime l'élément courant de la liste.

```
bool  DeleteCurrent()
```

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Note

Si la mode de gestion de la mémoire est activé, la mémoire est libérée.

Exemple :

```
//--- exemple d'utilisation de CList::DeleteCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   if(!list.DeleteCurrent())
     {
      printf("Delete error");
      delete list;
      return;
     }
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime l'élément de la liste situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans la liste de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Note

Si la mode de gestion de la mémoire est activé, la mémoire est libérée.

Exemple :

```
//--- exemple d'utilisation de CList::Delete(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   if(!list.Delete(0))
     {
      printf("Delete error");
      delete list;
      return;
     }
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments de la liste.

```
void  Clear()
```

Note

Si le mode de gestion de la mémoire est activé, les éléments supprimés sont exemptés.

Exemple :

```
//--- exemple d'utilisation de CList::Clear()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   //--- vide la liste
   list.Clear();
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

IndexOf

Retourne la position de l'élément dans la liste

```
int  IndexOf(
   CObject*  element      // Pointeur vers l'élément
   )
```

Paramètres

element

[in] Pointeur vers l'élément de la liste.

Valeur de Retour

Position de l'élément dans la liste ou -1.

Exemple :

```
//--- exemple d'utilisation de CList::IndexOf(CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   CObject *object=new CObject;
   if(object==NULL)
     {
      printf("Element create error");
      delete list;
      return;
     }
   if(list.Add(object))
     {
      int pos=list.IndexOf(object);
     }
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetNodeAtIndex

Retourne l'élément de la liste correspondant à la position spécifiée.

```
CObject*  GetNodeAtIndex(
   int  pos      // position
   )
```

Paramètres

pos

[in] Position de l'élément.

Valeur de retour

Pointeur sur l'élément en cas de succès, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetNodeAtIndex(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetNodeAtIndex(10);
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetFirstNode

Retourne le premier élément de la liste.

```
CObject*  GetFirstNode()
```

Valeur de Retour

Pointeur sur le premier élément, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetFirstNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetFirstNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetPrevNode

Retourne l'élément précédent de la liste.

```
CObject*  GetPrevNode()
```

Valeur de Retour

Pointeur sur l'élément précédent de la liste, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetPrevNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetPrevNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetCurrentNode

Retourne l'élément courant de la liste

```
CObject*  GetCurrentNode()
```

Valeur de Retour

Pointeur sur l'élément courant, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetCurrentNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetCurrentNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetNextNode

Retourne le prochain élément de la liste

```
CObject*  GetNextNode()
```

Valeur de Retour

Pointeur sur le prochain élément en cas de succès, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetNextNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetNextNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetLastNode

Retourne le dernier élément de la liste.

```
CObject*  GetLastNode()
```

Valeur de Retour

Pointeur sur le dernier élément, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetLastNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetLastNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Sort

Trie une liste.

```
void  Sort(
   int  mode      // Mode de tri
   )
```

Paramètres

mode

[in] Mode de tri.

Valeur de Retour

Aucune.

Note

Le tri de la liste est toujours effectué en mode ascendant.

Exemple :

```
//--- exemple d'utilisation de CList::Sort(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- Trie avec le mode 0
   list.Sort(0);
   //--- utilisation de la liste
   //--- ...
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

MoveToIndex

Déplace l'élément courant de la liste à la position spécifiée.

```
bool  MoveToIndex(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position de l'élément de la liste à déplacer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CList::MoveToIndex(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- déplacer le noeud courant pour commencer
   list.MoveToIndex(0);
   //--- utilisation de la liste
   //--- . . .
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Exchange

Intervertit deux éléments de la liste.

```
bool  Exchange(
   CObject*  node1,      // Premier élément de la liste
   CObject*  node2       // Deuxième élément de la liste
   )
```

Paramètres

node1

[in] Elément de la liste

node2

[in] Elément de la liste

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être intervertis.

Exemple :

```
//--- exemple d'utilisation de CList::Exchange(CObject*,CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- déplacement des éléments
   list.Exchange(list.GetFirstNode(),list.GetLastNode());
   //--- utilisation de la liste
   //--- . . .
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareList

Compare la liste avec une autre liste.

```
bool  CompareList(
   CList*  list      // Liste de comparaison
   )
```

Paramètres

list

[in] Pointeur vers une instance de la classe CList contenant les éléments sources à comparer.

Valeur de Retour

vrai si les listes sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CList::CompareList(const CList*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée la liste source
   CList *src=new CList;
   if(src==NULL)
     {
      printf("Object create error");
      delete list;
      return;
     }
   //--- remplissage de la liste source
   //--- . . .
   //--- comparaison avec une autre liste
   bool result=list.CompareList(src);
   //--- supprime la listes
   delete src;
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Search

Cherche un élément égal au modèle donné dans une liste triée.

```
CObject*  Search(
   CObject*  element      // Modèle
   )
```

Paramètres

element

[in] Modèle à rechercher dans la liste.

Valeur de Retour

Pointeur sur l'élément trouvé en cas de succès, NULL si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CList::Search(CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   //--- trie la liste
   list.Sort(0);
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete list;
      return;
     }
   //--- définit les attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(list.Search(sample)!=NULL) printf("Element found");
   else                          printf("Element not found");
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les données de la liste dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier précédemment ouvert avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CList::Save(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- supprime la liste
   delete list;
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données de la liste depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier précédemment ouvert avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

Lors de la lecture du fichier, chaque élément de la liste est créé en appelant la méthode CList::CreateElement().

Exemple :

```
//--- exemple d'utilisation de CLoad::Load(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- utilisation des éléments de la liste
   //--- . . .
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de la liste.

```
virtual int  Type()
```

Valeur de Retour

Identifant du type de la liste (7779 pour CList).

Exemple :

```
//--- exemple d'utilisation de CList::Type()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type de la liste
   int type=list.Type();
   //--- supprime la liste
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

CTreeNode

La classe CTreeNode représente un noeud de l'arbre binaire CTree.

Description

La classe CTreeNode permet de travailler avec les noeuds d'un arbre binaire CTree. Les options de navigation au travers de l'arbre sont implémentées dans la classe. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CTreeNode : public CObject
```

Titre

```
#include <Arrays\TreeNode.mqh>
```

Hiérarchie d'héritage

CObject

CTreeNode

Descendants directs

CTree

Méthodes de Classe

Attributs	
Owner	Retourne/Définit le pointeur du noeud père
Left	Retourne/Définit le pointeur situé à gauche
Right	Retourne/Définit le pointeur situé à droite
Balance	Retourne la balance du noeud
BalanceL	Retourne la balance de la sous-branche de gauche du noeud
BalanceR	Retourne la balance de la sous-branche de droite du noeud
Création d'u nouvel élément	
CreateSample	Création d'un nouveau noeud
Comparaison	
RefreshBalance	Recalcule la balance du noeud
Recherche	
GetNext	Retourne le pointeur du noeud suivant
Entrée/Sortie	

© 2000-2024, MetaQuotes Ltd.

Attributs	
SaveNode	Sauvegarde les données du noeud dans un fichier
LoadNode	Charge les données d'un noeud depuis un fichier
virtual Type	Retourne l'identifiant du type du noeud

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Compare

Les arbres de classes dérivant de CTreeNode sont mis en application.

Une classe dérivant de la classe doit implémenter les méthodes : CreateSample pour créer une nouvelle instance de la classe dérivant de la classe CTreeNode, Compare pour comparer les valeurs des champs clés de la classe dérivant de la classe CTreeNode, Type (si nécéssité d'identifier un noeud), SaveNode et LoadNode (si nécéssité de travailler avec des fichiers).

Considérons l'exemple d'une classe dérivée de la classe CTree.

```
//+------------------------------------------------------------------+
//|                                                   MyTreeNode.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                       https://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\TreeNode.mqh>
//+------------------------------------------------------------------+
//| Décrit la classe dérivée de CTreeNode. |
//+------------------------------------------------------------------+
//| Class CMyTreeNode. |
//| But : Classe d'un élément d'un arbre binaire. |
//|             Descendant de la classe CTreeNode. |
//+------------------------------------------------------------------+
class CMyTreeNode : public CTreeNode
  {
protected:
   //--- données utlisateur
   long              m_long;            // champ clé de type long
   double            m_double;          // variable personnelle de type double
   string            m_string;          // variable personnelle de type string
   datetime          m_datetime;        // variable personnelle de type datetime

public:
                     CMyTreeNode();
   //--- méthodes d'accès aux données utilisateur
   long              GetLong(void)                 { return(m_long); }
   void              SetLong(long value)           { m_long=value;   }
```

```
   double            GetDouble(void)              { return(m_double); }
   void              SetDouble(double value)      { m_double=value;  }
   string            GetString(void)              { return(m_string); }
   void              SetString(string value)      { m_string=value;  }
   datetime          GetDateTime(void)            { return(m_datetime); }
   void              SetDateTime(datetime value)  { m_datetime=value;  }
   //--- méthodes de travail avec les fichiers
   virtual bool      Save(int file_handle);
   virtual bool      Load(int file_handle);
protected:
   virtual int       Compare(const CObject *node,int mode);
   //--- méthodes de création des instances de classe
   virtual CTreeNode* CreateSample();
  };
//+------------------------------------------------------------------+
//| Constructeur de la classe CMyTreeNode. |
//| ENTREE :  aucun. |
//| SORTIE : aucune. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
void CMyTreeNode::CMyTreeNode()
  {
//--- initialisation des données utilisateur
   m_long        =0;
   m_double      =0.0;
   m_string      ="";
   m_datetime    =0;
  }
//+------------------------------------------------------------------+
//| Comparaison avec un autre noeud par l'algorithme spécifié. |
//| ENTREE :  noeud - élément à comparer,                        |
//|         mode - identifiant de l'algorithme de comparaison. |
//| SORTIE : résultat de la comparaison (>0,0,<0). |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
int CMyTreeNode::Compare(const CObject *node,int mode)
  {
//--- le paramètre mode est ignoré, car l'algorithme de construction de l'arbre est le
   int res=0;
//--- cast explicite
   CMyTreeNode *n=node;
   res=(int)(m_long-n.m_long);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Création d'une nouvelle instance de classe. |
//| ENTREE :  aucun. |
//| SORTIE : pointeur sur la nouvelle instance de classe CMyTreeNode. |
```

© 2000-2024, MetaQuotes Ltd.

```
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
CTreeNode* CMyTreeNode::CreateSample()
  {
   CMyTreeNode *result=new CMyTreeNode;
//---
   return(result);
  }
//+------------------------------------------------------------------+
//| Sauvegarde les données du noeud de l'arbre dans un fichier. |
//| ENTREE :  file_handle - handle d'un fichier ouvert en écriture. |
//| SORTIE : vrai si OK, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CMyTreeNode::Save(int file_handle)
  {
   uint i=0,len;
//--- vérifications préliminaires
   if(file_handle<0) return(false);
//--- écriture des données utilisateur
//--- écriture des variables personnelles de type long
   if(FileWriteLong(file_handle,m_long)!=sizeof(long))          return(false);
//--- écriture des variables personnelles de type double
   if(FileWriteDouble(file_handle,m_double)!=sizeof(double))    return(false);
//--- écriture des variables personnelles de type string
   len=StringLen(m_string);
//--- écriture de la longueur de la chaîne
   if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE)   return(false);
//--- écriture de la chaîne
   if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- écriture des données personnelles de type datetime
   if(FileWriteLong(file_handle,m_datetime)!=sizeof(long))      return(false);
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Charge les données du noeud de l'arbre depuis un fichier. |
//| ENTREE :  file_handle - handle d'un fichier ouvert en lecture. |
//| SORTIE : vrai si OK, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CMyTreeNode::Load(int file_handle)
  {
   uint i=0,len;
//--- vérifications préliminaires
   if(file_handle<0) return(false);
//--- lecture
   if(FileIsEnding(file_handle)) return(false);
//--- lecture des variables personnelles de type char
```

© 2000-2024, MetaQuotes Ltd.

```
//--- lecture des variables personnelles de type long
   m_long=FileReadLong(file_handle);
//--- lecture des variables personnelles de type double
   m_double=FileReadDouble(file_handle);
//--- lecture des variables personnelles de type string
//--- lecture de la longueur de la chaîne
   len=FileReadInteger(file_handle,INT_VALUE);
//--- lecture de la chaîne
   if(len!=0) m_string=FileReadString(file_handle,len);
   else       m_string="";
//--- lecture des variables personnelles de type datetime
   m_datetime=FileReadLong(file_handle);
//---
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

Owner

Retourne le pointeur du noeud père.

```
CTreeNode*  Owner()
```

Valeur de Retour

Pointeur sur le noeud père.

Owner

Définit le pointeur du noeud père.

```
void  Owner(
   CTreeNode*  node       // noeud
   )
```

Paramètres

node

[in] Nouvelle valeur du pointeur du noeud père.

Valeur de Retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Left

Retourne le pointeur du noeud situé à gauche.

```
CTreeNode*  Left()
```

Valeur de Retour

Pointeur du noeud situé à gauche.

Left

Définit le pointeur du noeud situé à gauche.

```
void  Left(
   CTreeNode*  node       // noeud
   )
```

Paramètres

node

[in] Nouvelle valeur du pointeur du noeud de gauche.

Valeur de Retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Right

Retourne le pointeur du noeud situé à droite.

```
CTreeNode*  Right()
```

Valeur de Retour

Le pointeur sur le noeud situé à droite.

Right

Définit le pointeur du noeud situé à droite.

```
void  Right(
   CTreeNode*  node       // noeud
   )
```

Paramètres

node

[in] Nouvelle valeur du pointeur du noeud de droite.

Valeur de Retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Balance

Retourne la balance du noeud.

```
int  Balance() const
```

Valeur de Retour

Balance du noeud.

© 2000-2024, MetaQuotes Ltd.

BalanceL

Retourne la balance de la sous-branche de gauche du noeud.

```
int  BalanceL() const
```

Valeur de Retour

Balance de la of the left sub-branch of the node.

© 2000-2024, MetaQuotes Ltd.

BalanceR

Retourne la balance de la sous-branche de droite du noeud.

```
int  BalanceR() const
```

Valeur de Retour

Balance de la sous-branche de droite du noeud.

© 2000-2024, MetaQuotes Ltd.

CreateSample

Crée un nouveau noeud modèle.

```
virtual CTreeNode*  CreateSample()
```

Valeur de Retour

Pointeur vers le nouveau noeud modèle ou NULL.

© 2000-2024, MetaQuotes Ltd.

RefreshBalance

Recalcule la balance du noeud.

```
int  RefreshBalance()
```

Valeur de Retour

Balance du noeud.

© 2000-2024, MetaQuotes Ltd.

GetNext

Retourne un pointeur vers le noeud suivant.

```
CTreeNode*  GetNext(
   CTreeNode*  node       // noeud
   )
```

Paramètres

node

[in] Noeud de départ de la recherche.

Valeur de Retour

Pointeur sur le noeud suivant.

© 2000-2024, MetaQuotes Ltd.

SaveNode

// Sauvegarde les données du noeud dans un fichier.

```
bool  SaveNode(
   int  file_handle      // handle
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire ouvert précédemment en écriture.

Valeur de Retour

vrai en cas de succès, faux sinon.

© 2000-2024, MetaQuotes Ltd.

LoadNode

// Charge les données du noeud depuis un fichier.

```
bool  LoadNode(
   int         file_handle,      // handle
   CTreeNode*  main              // noeud
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire ouvert précédemment en lecture.

main

[in] Noeud à remplir.

Valeur de Retour

vrai en cas de succès, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type du noeud.

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du noeud.

© 2000-2024, MetaQuotes Ltd.

CTree

La classe CTree est un arbre binaire d'instances de classe CTreeNode et de ses descendants.

Description

La classe CTree permet de travailler avec un arbre binaire d'instances de la classe CTreeNode et de ses descendants. Les méthodes d'ajout, insertion et suppression des éléments et de recherche dans l'arbre sont implémentées dans cette classe. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Noter que le mécanisme de gestion de la mémoire n'est pas implémenté dans la classe CTree (contrairement aux classes CList et CArrayObj). Tous les noeuds de l'arbre sont supprimés et la mémoire libérée.

Déclaration

```
class CTree : public CTreeNode
```

Titre

```
#include <Arrays\Tree.mqh>
```

Hiérarchie d'héritage

CObject
  CTreeNode
    CTree

Méthodes de Classe

Attributs	
Root	Retourne le noeud racine de l'arbre
Création d'u nouvel élément	
CreateElement	Création d'un nouveau noeud
Remplissage	
Insert	Ajout d'un noeud à un arbre
Suppression	
Detach	Détache de l'arbre un noeud spécifié
Delete	Supprime de l'arbre un noeud spécifié
Clear	Supprime tous les noeuds de l'arbre
Recherche	
Find	Recherche un noeud dans l'arbre à partir d'un modèle

© 2000-2024, MetaQuotes Ltd.

Attributs	
Entrée/Sortie	
virtual Save	Sauvegarde les données de l'arbre dans un fichier
virtual Load	Charge les données d'un arbre depuis un fichier
virtual Type	Retourne l'identifiant du type de l'arbre

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CTreeNode

Parent, Parent, Left, Left, Right, Right, Balance, BalanceL, BalanceR, RefreshBalance, GetNext, SaveNode, LoadNode

Les arbres de classes CTreeNode dérivant de CTree sont mis en application.

Les classes filles de la classe CTree doivent implémenter la méthode prédéfinie CreateElement qui crée un nouveau CTreeNode.

Considérons l'exemple d'une classe dérivée de la classe CTree.

```
//+------------------------------------------------------------------+
//|                                                       MyTree.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//| www.metaquotes.net |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\Tree.mqh>
#include "MyTreeNode.mqh"
//---
input int extCountedNodes = 100;
//+------------------------------------------------------------------+
//| Describe class CMyTree derived from CTree. |
//+------------------------------------------------------------------+
//| Class CMyTree.                                                   |
//| But : Construction et navigation d'une recherche binaire dans un arbre. |
//+------------------------------------------------------------------+
class CMyTree : public CTree
  {
public:
   //--- méthode de recherche dans l'arbre à partir de données spécifiques
   CMyTreeNode*        FindByLong(long find_long);
   //--- méthode de création de l'élément dans l'arbre
   virtual CTreeNode *CreateElement();
  };
//---
```

```
CMyTree MyTree;
//+-----------------------------------------------------------------+
//| Création d'un nouveau noeud dans l'arbre. |
//| ENTREE :  aucun. |
//| SORTIE : pointeur vers le nouveau noeud de l'arbre en cas de succès, sinon NULL.
//| REMARQUE : aucune. |
//+-----------------------------------------------------------------+
CTreeNode *CMyTree::CreateElement()
  {
   CMyTreeNode *node=new CMyTreeNode;
//---
   return(node);
  }
//+-----------------------------------------------------------------+
//| Cherche un élément à partir d'une valeur m_long. |
//| ENTREE :  find_long - valeur recherchée. |
//| SORTIE : pointeur sur l'élément trouvé, ou NULL. |
//| REMARQUE : aucune. |
//+-----------------------------------------------------------------+
CMyTreeNode* CMyTree::FindByLong(long find_long)
  {
   CMyTreeNode *res=NULL;
   CMyTreeNode *node;
//--- crée un nouveau noeud pour passer le paramètre de recherche
   node=new CMyTreeNode;
   if(node==NULL) return(NULL);
   node.SetLong(find_long);
//---
   res=Find(node);
   delete node;
//---
   return(res);
  }
//+-----------------------------------------------------------------+
//| script "test de la class CMyTree"                               |
//+-----------------------------------------------------------------+
//---  tableau de chaînes d'initialisation
string str_array[11]={"p","oo","iii","uuuu","yyyyy","ttttt","rrrr","eee","ww","q","999
//---
int OnStart() export
  {
   int          i;
   uint         pos;
   int          beg_time,end_time;
   CMyTreeNode *node; //--- pointeur temporaire vers le modèle de la classe CMyTreeNoc
//---
   printf("Start test %s.",__FILE__);
//--- Remplissage de MyTree avec les modèles de classe MyTreeNode.
   beg_time=GetTickCount();
```

```
   for(i=0;i<extCountedNodes;i++)
     {
      node=MyTree.CreateElement();
      if(node==NULL)
        {
         //--- sortie d'urgence
         printf("%s (%4d): create error",__FILE__,__LINE__);
         return(__LINE__);
        }
      NodeSetData(node,i);
      node.SetLong(i);
      MyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("Filling time of MyTree is %d ms.",end_time-beg_time);
//--- Crée un arbre temporaire TmpMyTree.
   CMyTree TmpMyTree;
//--- Détache 50% des éléments de l'arbre (tous les noeuds ayant une valeur paire)
//--- et les ajoute à la table temporaire TmpMyTree.
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i+=2)
     {
      node=MyTree.FindByLong(i);
      if(node!=NULL)
         if(MyTree.Detach(node)) TmpMyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("Deletion time of %d elements from MyTree is %d ms.",extCountedNodes/2,end_t
//--- Retourne l'élément détaché
   node=TmpMyTree.Root();
   while(node!=NULL)
     {
      if(TmpMyTree.Detach(node)) MyTree.Insert(node);
      node=TmpMyTree.Root();
     }
//--- Appelle et vérifie la méthode Save(int file_handle);
   int file_handle;
   file_handle=FileOpen("MyTree.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!MyTree.Save(file_handle))
        {
         //--- erreur d'écriture dans le fichier
         //--- sortie d'urgence
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- fermeture du fichier avant de quitter !!!
         FileClose(file_handle);
         return(__LINE__);
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      FileClose(file_handle);
     }
//--- Appelle et vérifie la méthode Load(int file_handle);
   file_handle=FileOpen("MyTree.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!TmpMyTree.Load(file_handle))
        {
         //--- erreur de lecture du fichier
         //--- sortie d'urgence
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- fermeture du fichier avant de quitter !!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
     }
//---
   MyTree.Clear();
   TmpMyTree.Clear();
//---
   printf("End test %s. OK!",__FILE__);
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| Fonction de journalisation du contenu d'un noeud |
//+------------------------------------------------------------------+
void NodeToLog(CMyTreeNode *node)
  {
   printf("   %I64d,%f,'%s','%s'",
               node.GetLong(),node.GetDouble(),
               node.GetString(),TimeToString(node.GetDateTime()));
  }
//+------------------------------------------------------------------+
//| Fonction de "remplissage" d'un noeud avec des valeurs aléatoires |
//+------------------------------------------------------------------+
void NodeSetData(CMyTreeNode *node,int mode)
  {
   if(mode%2==0)
     {
      node.SetLong(mode*MathRand());
      node.SetDouble(MathPow(2.02,mode)*MathRand());
     }
   else
     {
      node.SetLong(mode*(long)(-1)*MathRand());
      node.SetDouble(-MathPow(2.02,mode)*MathRand());
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   node.SetString(str_array[mode%10]);
   node.SetDateTime(10000*mode);
  }
```

© 2000-2024, MetaQuotes Ltd.

Root

Retourne le noeud racine de l'arbre.

```
CTreeNode*  Root() const
```

Valeur de Retour

Pointeur vers le noeud racine de l'arbre.

© 2000-2024, MetaQuotes Ltd.

CreateElement

Crée un nouvelle instance de noeud.

```
virtual CTreeNode*  CreateElement()
```

Valeur de Retour

Pointeur vers la nouvelle instance du noeud ou NULL.

© 2000-2024, MetaQuotes Ltd.

Insert

Ajoute un noeud à l'arbre.

```
CTreeNode*  Insert(
   CTreeNode*  new_node       // noeud
   )
```

Paramètres

new_node

[in] pointeur vers le noeud à insérer dans l'arbre.

Valeur de Retour

Pointeur sur le noeud père ou NULL.

© 2000-2024, MetaQuotes Ltd.

Detach

Détache un noeud spécifié de l'arbre.

```
bool  Detach(
   CTreeNode*  node       // noeud
   )
```

Paramètres

node

[in] Pointeur vers le noeud à détacher.

Valeur de Retour

vrai en cas de succès, faux sinon.

Note

Après avoir détaché le noeud, le pointeur n'est pas libéré. L'arbre est équilibré.

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime de l'arbre un noeud spécifié.

```
bool  Delete(
   CTreeNode*  node       // noeud
   )
```

Paramètres

node

[in] Pointeur vers le noeud à supprimer.

Valeur de Retour

vrai en cas de succès, faux sinon.

Note

Après la suppression du noeud, le pointeur est libéré. L'arbre est équilibré.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les noeuds de l'arbre.

```
void  Clear()
```

Valeur de Retour

Aucune.

Note

Après la suppression des noeuds, tous les pointeurs sont libérés.

© 2000-2024, MetaQuotes Ltd.

Find

Recherche un noeud dans l'arbre à partir d'un modèle.

```
CTreeNode*  Find(
   CTreeNode*  node       // noeud
   )
```

Paramètres

node

[in] Noeud contenant les données à chercher.

Valeur de Retour

Pointeurr sur le noeud s'il a été trouvé, NULL sinon.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les données de l'arbre dans un fichier.

```
virtual bool  Save(
   int  file_handle      // handle
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire ouvert précédemment en écriture.

Valeur de Retour

vrai en cas de succès, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les données d'un arbre depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // handle
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire ouvert précédemment en lecture.

Valeur de Retour

vrai en cas de succès, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'arbre.

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type de l'arbre.

© 2000-2024, MetaQuotes Ltd.

Collections de Données Génériques

La bibliothèque fournit les classes et les interfaces qui définissent les collections génériques, permettant aux utilisateurs de créer des collections fortement typées. Ces collections sont plus pratiques et efficaces dans la gestion de leurs données que des collections typées non génériques.

La bibliothèque est disponible dans le dossier Include\Generic du répertoire de travail du terminal.

Objets :

Objet	Description	Type
ICollection	Interface pour implémenter des collections de données génériques	INTERFACE
IEqualityComparable	Interface pour implémenter des objets qui peuvent être comparés	INTERFACE
IComparable	Interface pour implémenter des objets qui peuvent être comparés en terme de "plus grand que, inférieur ou égal à"	INTERFACE
IComparer	Interface pour implémenter une classe générique qui compare 2 objets de type T et vérifie si l'un est "plus grand que, inférieur ou égal au" second objet	INTERFACE
IEqualityComparer	Interface pour implémenter une classe générique qui compare et vérifie l'égalité de 2 objets de type T	INTERFACE
IList	Interface pour implémenter les listes de données génériques	INTERFACE
IMap	Interface pour implémenter les collections génériques de paires clé/valeur	INTERFACE
ISet	Interface pour implémenter les ensembles (set) de données génériques	INTERFACE
CDefaultComparer	Une classe d'aide qui implémente l'interface générique IComparer<T> basée sur les méthodes globales Compare	CLASS
CDefaultEqualityComparer	Une classe d'aide qui implémente l'interface générique IEqualityComparer<T> en utilisant les méthodes globales Equals<T> et GetHashCode	CLASS

© 2000-2024, MetaQuotes Ltd.

Objet	Description	Type
CArrayList	Une classe générique qui implémente l'interface IList<T>	CLASS
CKeyValuePair	La classe d'implémentation d'une paire clé/valeur	CLASS
CHashMap	Une classe générique qui implémente l'interface IMap<TKey, TValue>	CLASS
CHashSet	Une classe générique qui implémente l'interface ISet<T>	CLASS
CLinkedListNode	Une classe d'aide pour implémenter la classe CLinkedListNode<T>	CLASS
CLinkedList	Une classe générique qui implémente l'interface ICollection<T>	CLASS
CQueue	Une classe générique qui implémente l'interface ICollection<T>	CLASS
CRedBlackTreeNode	Une classe d'aide utilisée dans l'implémentation de la classe CRedBlackTree<T>	CLASS
CRedBlackTree	Une classe générique qui implémente l'interface ICollection<T>	CLASS
CSortedMap	Une classe générique qui implémente l'interface IMap<TKey, TValue>	CLASS
CSortedSet	Une classe générique qui implémente l'interface ISet<T>	CLASS
CStack	Une classe générique qui implémente l'interface ICollection<T>	CLASS

Méthodes globales :

Méthode	Description
ArrayBinarySearch	Recherche la valeur spécifiée dans un tableau unidimensionnel trié de façon ascendante en utilisant l'interface IComparable<T> pour comparer les éléments
ArrayIndexOf	Recherche la première occurrence d'une valeur dans un tableau à 1 dimension
ArrayLastIndexOf	Recherche la dernière occurrence d'une valeur dans un tableau à 1 dimension
ArrayReverse	Change l'ordre des éléments dans un tableau à 1 dimension

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
Compare	Compare 2 valeurs, que l'une soit supérieure, inférieure ou égale à l'autre
Equals	Compare l'égalité de 2 valeurs
GetHashCode	Calcule la valeur du code de hashage

© 2000-2024, MetaQuotes Ltd.

ICollection<T>

ICollection<T> est une interface pour implémenter les collections de données génériques.

Description

L'interface ICollection<T> détermine les méthodes de base pour travailler avec le collections, incluant les méthodes pour compter les éléments, pour vider une collection, pour ajouter ou supprimer des éléments, etc.

Déclaration

```
template<typename T>
interface ICollection
```

Fichier header

```
#include <Generic\Interfaces\ICollection.mqh>
```

Hiérarchie d'Héritage

ICollection

Descendants directs

CLinkedList, CQueue, CRedBlackTree, CStack, IList, IMap, ISet

Méthodes de classe

Méthode	Description
Add	Ajoute un élément à une collection
Count	Retourne le nombre d'éléments dans une collection
Contains	Détermine si une collection contient un élément ayant la valeur spécifiée
CopyTo	Copie tous les éléments d'une collection dans le tableau spécifié en commençant à l'indice spécifié
Clear	Supprime tous les éléments d'une collection
Remove	Supprime la première occurrence de l'élément spécifié dans une collection

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à une collection.

```
bool Add(
   T  value     // la valeur de l'élément
   );
```

Paramètres

value

[in] La valeur de l'élément à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Count

Retourne le nombre d'éléments dans une collection.

```
int Count();
```

Valeur de Retour

Retourne le nombre d'éléments.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si une collection contient un élément ayant la valeur spécifiée.

```
bool Contains(
   T  item     // la valeur à chercher
   );
```

Paramètres

item

[in] La valeur à chercher.

Valeur de Retour

Retourne true si un élément ayant la valeur spécifié a été trouvé dans la collection, false sinon.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie tous les éléments d'une collection dans le tableau spécifié en commençant à l'indice spécifié.

```
int CopyTo(
   T&          dst_array[],      // le tableau de destination
   const int   dst_start=0       // indice de début d'écriture
   );
```

Paramètres

&dst_array[]

[out] Un tableau dans lequel les éléments de la collection seront écrits.

dst_start=0

[in] Un indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre d'éléments copiés.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments d'une collection.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime la première occurrence de l'élément spécifié dans une collection.

```
bool Remove(
   T  item     // l'élément par valeur
   );
```

Paramètres

item

[in] La valeur de l'élément à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

IEqualityComparable<T>

IEqualityComparable<T> est une interface pour implémenter des objets qui peuvent être comparés.

Description

L'interface IEqualityComparable<T> définit les méthodes pour calculer le code de hashage de l'objet courant et pour vérifier s'il est égal à un autre objet du même type.

Déclaration

```
template<typename T>
interface IEqualityComparable
```

Fichier header

```
#include <Generic\Interfaces\IEqualityComparable.mqh>
```

Hiérarchie d'Héritage

IEqualityComparable

Descendants directs

IComparable

Méthodes de classe

Méthode	Description
Equals	Compare l'objet courant avec la valeur spécifiée
HashCode	Calcule la valeur du code de hashage de l'objet courant

© 2000-2024, MetaQuotes Ltd.

Equals

Compare l'objet courant avec la valeur spécifiée.

```
bool Equals(
   T  value     // la valeur à comparer
   );
```

Paramètres

value

[in] La valeur avec laquelle l'objet courant doit être comparé.

Valeur de Retour

Retourne true si les objets sont égaux, false sinon.

© 2000-2024, MetaQuotes Ltd.

HashCode

Calcule la valeur du code de hashage de l'objet courant.

```
int HashCode();
```

Valeur de Retour

Retourne le code de hashage.

© 2000-2024, MetaQuotes Ltd.

IComparable<T>

IComparable<T> est une interface pour implémenter des objets pouvant être comparés pour déterminer si l'un est plus grand, égal ou plus petit que l'autre

Description

L'interface IComparable<T> définit une méthode pour comparer l'objet courant avec un autre objet du même type, grâce à quoi la collection de ces objets peut être triée.

Déclaration

```
template<typename T>
interface IComparable : public IEqualityComparable<T>
```

Fichier header

```
#include <Generic\Interfaces\IComparable.mqh>
```

Hiérarchie d'Héritage

IEqualityComparable

IComparable

Descendants directs

CKeyValuePair

Méthodes de classe

Méthode	Description
Compare	Compare l'objet courant avec la valeur spécifiée

© 2000-2024, MetaQuotes Ltd.

Compare

Compare l'objet courant avec la valeur spécifiée.

```
int Compare(
   T  value     // la valeur à comparer
   );
```

Paramètres

value

[in] La valeur avec laquelle l'objet courant doit être comparé.

Valeur de Retour

Retourne un nombre exprimant le rapport de l'objet courant avec l'objet passé :

- si le résultat est inférieur à 0, l'objet courant est plus petit que l'objet passé en argument
- si le résultat est égal à 0, les deux objets sont égaux
- si le résultat est supérieur à 0, l'objet courant est plus grand que l'objet passé en argument

© 2000-2024, MetaQuotes Ltd.

IComparer<T>

IComparer<T> est une interface pour implémenter une classe générique comparant 2 objets de type T, pour déterminer si l'un est plus grand, égal ou plus petit que l'autre

Description

L'interface IComparer<T> définit une méthode pour comparer 2 objets de type T, grâce à quoi la collection de ces objets peut être triée.

Déclaration

```
template<typename T>
interface IComparer
```

Fichier header

```
#include <Generic\Interfaces\IComparer.mqh>
```

Hiérarchie d'Héritage

IComparer

Descendants directs

CDefaultComparer

Méthodes de classe

Méthode	Description
Compare	Compare 2 valeurs de type T

© 2000-2024, MetaQuotes Ltd.

Compare

Compare 2 valeurs de type T.

```
int Compare(
   T  x,      // la première valeur
   T  y       // la deuxième valeur
   );
```

Paramètres

x

[in] La première valeur à comparer.

y

[in] La première valeur à comparer.

Valeur de Retour

Retourne un nombre qui exprime le ratio de 2 valeurs comparées :

- si le résultat est inférieur à 0, x est inférieur à y (x<y)
- si le résultat est égal à 0, x est égal à y (x=y)
- si le résultat est supérieur à 0, x est plus grand que y (x>y)

© 2000-2024, MetaQuotes Ltd.

IEqualityComparer<T>

IEqualityComparer<T> est une interface pour implémenter une classe générique qui compare 2 objets de type T.

Description

L'interface IEqualityComparer<T> définit les méthodes pour calculer le code de hashage d'un objet de type T et pour vérifier si 2 objets de type T sont égaux.

Déclaration

```
template<typename T>
interface IEqualityComparer
```

Fichier header

```
#include <Generic\Interfaces\IEqualityComparer.mqh>
```

Hiérarchie d'Héritage

IEqualityComparer

Descendants directs

CDefaultEqualityComparer

Méthodes de classe

Méthode	Description
Equals	Compare 2 valeurs de type T
HashCode	Calcule la valeur du code de hashage suivant l'objet de type T

© 2000-2024, MetaQuotes Ltd.

Equals

Compare 2 valeurs de type T.

```
bool Equals(
   T  x,      // la première valeur
   T  y       // la deuxième valeur
   );
```

Paramètres

x

[in] La première valeur à comparer.

y

[in] La deuxième valeur à comparer.

Valeur de Retour

Retourne true si les valeurs sont égales, false sinon.

© 2000-2024, MetaQuotes Ltd.

HashCode

Calcule la valeur du code de hashage suivant l'objet de type T.

```
int HashCode(
   T  value     // un objet pour le calcul
   );
```

Paramètres

value

[in] L'objet pour lequel vous désirez obtenir le code de hashage.

Valeur de Retour

Retourne le code de hashage.

© 2000-2024, MetaQuotes Ltd.

IList<T>

IList<T> est une interface pour implémenter les listes de données génériques.

Description

L'interface IList<T> définit les méthodes de base pour travailler avec les listes, telles que l'accès à un élément par sa position, la recherche et la suppression d'éléments, le tri, etc.

Déclaration

```
template<typename T>
interface IList : public ICollection<T>
```

Fichier header

```
#include <Generic\Interfaces\IList.mqh>
```

Hiérarchie d'Héritage

ICollection

IList

Descendants directs

CArrayList

Méthodes de classe

Méthode	Description
TryGetValue	Retourne l'élément de la liste à la position spécifiée
TrySetValue	Change une valeur de la liste à l'indice spécifié
Insert	Insère un élément dans la liste à l'indice spécifié
IndexOf	Recherche la première occurrence d'une valeur dans une liste
LastIndexOf	Recherche la dernière occurrence d'une valeur dans une liste
RemoveAt	Supprime l'élément de la liste à l'indice spécifié

© 2000-2024, MetaQuotes Ltd.

TryGetValue

Retourne l'élément de la liste à la position spécifiée.

```
bool TryGetValue(
   const int  index,      // indice de l'élément
   T&         value       // une variable pour écrire la valeur
   );
```

Paramètres

index

[in] L'indice de l'élément dans la liste.

&value

[out] La variable dans laquelle la valeur spécifiée de l'élément de la liste sera écrite.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

TrySetValue

Change une valeur de la liste à l'indice spécifié.

```
bool TrySetValue(
   const int  index,      // indice de l'élément
   T          value       // nouvelle valeur
   );
```

Paramètres

index

[in] L'indice de l'élément dans la liste.

value

[in] La nouvelle valeur à assigner à l'élémet spécifié de la liste.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Insert

Insère un élément dans la liste à l'indice spécifié.

```
bool Insert(
   const int  index,      // indice où insérer
   T          item        // la valeur à insérer
   );
```

Paramètres

index

[in] l'indiceoù la valeur sera insérée.

item

[in] La valeur à insérer à l'indice spécifié.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

IndexOf

Recherche la première occurrence d'une valeur dans une liste.

```
int IndexOf(
   T  item     // la valeur à chercher
   );
```

Paramètres

item

[in] La valeur à chercher.

Valeur de Retour

Retourne l'indice du premier élément trouvé. Si la valeur n'est pas trouvée, retourne -1.

© 2000-2024, MetaQuotes Ltd.

LastIndexOf

Recherche la dernière occurrence d'une valeur dans une liste.

```
int LastIndexOf(
   T  item     // la valeur à chercher
   );
```

Paramètres

item

[in] La valeur à chercher.

Valeur de Retour

Retourne l'indice du dernier élément trouvé. Si la valeur n'est pas trouvée, retourne -1.

© 2000-2024, MetaQuotes Ltd.

RemoveAt

Supprime l'élément de la liste à l'indice spécifié.

```
bool RemoveAt(
   const int  index     // indice de l'élément
   );
```

Paramètres

index

[in] L'indice de l'élément à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

IMap<TKey, TValue>

IMap<TKey, TValue> est une interface pour implémenter des collections génériques de paires clé/valeur.

Description

L'interface IMap<TKey, TValue> définit les méthodes de base pour travailler avec les collections dont les données sont stockées sous forme de paires clé/valeur.

Déclaration

```
template<typename TKey, typename TValue>
interface IMap : public ICollection<TKey>
```

Fichier header

```
#include <Generic\Interfaces\IMap.mqh>
```

Hiérarchie d'Héritage

ICollection

IMap

Descendants directs

CHashMap, CSortedMap

Méthodes de classe

Méthode	Description
Add	Ajoute une paire clé/valeur dans une collection
Contains	Détermine si une collection contient la paire clé/valeur avec la clé spécifiée
Remove	Supprime la première occurrence de la paire clé/valeur d'une collection
TryGetValue	Retourne l'élément ayant la clé spécifiée d'une collection
TrySetValue	Change la valeur d'une paire clé/valeur dans une collection ayant la clé spécifiée
CopyTo	Copie toutes les paires clé/valeur d'une collection vers les tableaux spécifiés, en commençant à l'indice spécifié

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute une paire clé/valeur dans une collection.

```
bool Add(
   TKey    key,       // clé
   TValue  value      // valeur
   );
```

Paramètres

key

[in] Clé.

value

[in] Valeur.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si une collection contient la paire clé/valeur ayant la clé spécifiée.

```
bool Contains(
   TKey    key,       // clé
   TValue  value      // valeur
   );
```

Paramètres

key

[in] Clé.

value

[in] Valeur.

Valeur de Retour

Retourne true si une collection contient la paire clé/valeur avec les clé et valeur spécifiées, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime la première occurrence de la paire clé/valeur d'une collection.

```
bool Remove(
   TKey  key     // clé
   );
```

Paramètres

key

[in] Clé.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

TryGetValue

Retourne l'élément d'une collection ayant la clé spécifiée.

```
bool TryGetValue(
   TKey     key,       // clé
   TValue&  value      // une variable pour écrire la valeur
   );
```

Paramètres

key

[in] Clé.

&value

[out] La variable dans laquelle la valeur spécifiée de la paire clé/valeur sera écrite.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

TrySetValue

Change la valeur de la paire clé/valeur de la collection ayant la clé spécifiée.

```
bool TrySetValue(
   TKey    key,       // clé
   TValue  value      // nouvelle valeur
   );
```

Paramètres

key

[in] Clé.

value

[in] La nouvelle valeur à assigner à la paire clé/valeur spécifiée.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie toutes les paires clé/valeur d'une collection vers les tableaux spécifiés, en commençant à l'indice spécifié.

```
int CopyTo(
   TKey&      dst_keys[],       // le tableau de destination keys
   TValue&    dst_values[],     // le tableau de destination values
   const int  dst_start=0       // l'indice de départ for writing
   );
```

Paramètres

&dst_keys[]

[out] Un tableau vers lequel toutes les clés de la collection seront écrites.

&dst_values[]

[out] Un tableau vers lequel toutes les clés correspondantes de la collection seront écrites.

dst_start=0

[in] Un indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre de paires clé/valeurs copiées.

© 2000-2024, MetaQuotes Ltd.

ISet<T>

ISet<T> est une interface pour implémenter les sets de données génériques.

Description

L'interface ISet fournit les méthodes de base pour travailler avec les sets, telles que : l'union et l'intersection des sets, la définition de sous-ensembles stricts et non stricts, etc.

Déclaration

```
template<typename T>
interface ISet : public ICollection<T>
```

Fichier header

```
#include <Generic\Interfaces\ISet.mqh>
```

Hiérarchie d'Héritage

ICollection

ISet

Descendants directs

CHashSet, CSortedSet

Méthodes de classe

Méthode	Description
ExceptWith	Effectue l'opération de différence entre la collection courante et la collection passée en argument (tableau)
IntersectWith	Effectue l'opération d'intersection entre la collection courante et la collection passée en argument (tableau)
SymmetricExceptWith	Effectue l'opération de différence symétrique entre la collection courante et la collection passée en argument (tableau)
UnionWith	Effectue l'union entre la collection courante et la collection passée en argument (tableau)
IsProperSubsetOf	Détermine si le set actuel est un vrai sous-ensemble de la collection ou du tableau spécifié
IsProperSupersetOf	Détermine si le set actuel est un vrai sur-ensemble de la collection ou du tableau spécifié
IsSubsetOf	Détermine si le set actuel est un sous-ensemble de la collection ou du tableau spécifié
IsSupersetOf	Détermine si le set actuel est un sur-ensemble de la collection ou du tableau spécifié

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
Overlaps	Détermine si le set actuel chevauche la collection ou le tableau spécifié
SetEquals	Détermine si le set actuel contient tous les éléments de la collection ou du tableau spécifié

© 2000-2024, MetaQuotes Ltd.

ExceptWith

Effectue l'opération de différence entre la collection courante et la collection passée en argument (tableau). Elle enlève de la collection courante (tableau) tous les éléments qui se trouvent dans la collection spécifiée (tableau).

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void ExceptWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection à sortir du set actuel.

&collection[]

[in] Un tableau à sortir du set actuel.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

IntersectWith

Effectue l'opération d'intersection entre la collection courante et la collection passée en argument (tableau). Elle modifie la collection courante pour ne contenir que les éléments qui se trouvent dans la collection spécifiée (tableau).

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void IntersectWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection avec laquelle opérer l'intersection avec le set actuel.

&collection[]

[in] Un tableau avec lequel opérer l'intersection avec le set actuel.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

SymmetricExceptWith

Effectue l'opération de différence symétrique entre la collection courante et la collection passée en argument (tableau). Elle modifie la collection courante pour ne contenir que les éléments qui se trouvent dans l'objet source ou dans la collection spécifiée (tableau) mais pas dans les deux.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void SymmetricExceptWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection avec laquelle produire la différence symétrique.

&collection[]

[in] Un tableau avec lequel produire la différence symétrique.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

UnionWith

Effectue l'union entre la collection courante et la collection passée en argument (tableau). Elle ajoute les éléments manquants de la collection spécifiée (tableau) dans la collection courante (array).

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void UnionWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection avec laquelle opérer l'union avec le set actuel.

&collection[]

[in] Un tableau avec lequel opérer l'union avec le set actuel.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

IsProperSubsetOf

Détermine si le set actuel est un vrai sous-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsProperSubsetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set courant est un vrai sous-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

IsProperSupersetOf

Détermine si le set actuel est un vrai sur-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsProperSupersetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set courant est un vrai sur-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

IsSubsetOf

Détermine si le set actuel est un sous-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsSubsetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set courant est un sous-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

IsSupersetOf

Détermine si le set actuel est un sur-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsSupersetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set courant est un sur-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

Overlaps

Détermine si le set actuel chevauche la collection ou le tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // une collection à comparer
   );
```

Version utilisant un tableau.

```
bool Overlaps(
   T&  array[]                     // un tableau à comparer
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer le chevauchement.

&collection[]

[in] Un tableau pour déterminer le chevauchement.

Valeur de Retour

Retourne true si le set actuel et une collection ou un tableau se chevauchent, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

SetEquals

Détermine si le set actuel contient tous les éléments de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // une collection à comparer
   );
```

Version utilisant un tableau.

```
bool SetEquals(
   T&  array[]                     // un tableau à comparer
   );
```

Paramètres

**collection*

[in] Une collection d'éléments à comparer.

&collection[]

[in] Une collection d'éléments à comparer.

Valeur de Retour

Retourne true si le set actuel contient tous les éléments de la collection ou du tableau spécifié, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

CDefaultComparer<T>

CDefaultComparer<T> est une classe d'aide qui implémente l'interface générique IComparer<T> basée sur les méthodes globales Compare.

Description

La classe CDefaultComparer<T> est utilisée par défaut dans les collections de données génériques, sauf si l'utilisateur utilise implicitement une autre classe implémentant l'interface IComparer<T>.

Déclaration

```
template<typename T>
class CDefaultComparer : public IComparer<T>
```

Fichier header

```
#include <Generic\Internal\DefaultComparer.mqh>
```

Hiérarchie d'Héritage

IComparer
  CDefaultComparer

Méthodes de classe

Méthode	Description
Compare	Compare 2 valeurs de type T

© 2000-2024, MetaQuotes Ltd.

Compare

Compare 2 valeurs de type T.

```
int Compare(
   T  x,      // la première valeur
   T  y       // la deuxième valeur
   );
```

Paramètres

x

[in] La première valeur à comparer.

y

[in] La deuxième valeur à comparer.

Valeur de Retour

Retourne un nombre qui exprime le ratio de 2 valeurs comparées :

- si le résultat est inférieur à 0, x est inférieur à y (x<y)
- si le résultat est égal à 0, x est égal à y (x=y)
- si le résultat est supérieur à 0, x est plus grand que y (x>y)

Note

Les valeurs x et y sont comparées suivant l'une des surcharges de la méthode globale Compare suivant le type T.

© 2000-2024, MetaQuotes Ltd.

CDefaultEqualityComparer<T>

CDefaultEqualityComparer<T> est une classe d'aide qui implémente l'interface générique IEqualityComparer<T> basée sur les méthodes globales Equals<T> et GetHashCode.

Description

La classe CDefaultEqualityComparer<T> est utilisée par défaut dans les collections de données génériques, sauf si l'utilisateur utilise implicitement une autre classe implémentant l'interface IEqualityComparer<T>.

Déclaration

```
template<typename T>
class CDefaultEqualityComparer : public IEqualityComparer<T>
```

Fichier header

```
#include <Generic\Internal\DefaultEqualityComparer.mqh>
```

Hiérarchie d'Héritage

IEqualityComparer
  CDefaultEqualityComparer

Méthodes de classe

Méthode	Description
Equals	Compare 2 valeurs de type T
HashCode	Calcule la valeur du code de hashage suivant l'objet de type T

© 2000-2024, MetaQuotes Ltd.

Equals

Compare 2 valeurs de type T.

```
bool Equals(
   T  x,      // la première valeur
   T  y       // la deuxième valeur
   );
```

Paramètres

x

[in] La première valeur à comparer.

y

[in] La deuxième valeur à comparer.

Valeur de Retour

Retourne true si les valeurs sont égales, false sinon.

© 2000-2024, MetaQuotes Ltd.

HashCode

Calcule la valeur du code de hashage suivant l'objet de type T.

```
int HashCode(
   T  value     // l'objet pour le calcul
   );
```

Paramètres

value

[in] L'objet pour lequel vous désirez obtenir le code de hashage.

Valeur de Retour

Retourne le code de hashage.

© 2000-2024, MetaQuotes Ltd.

CRedBlackTreeNode<T>

CRedBlackTreeNode<T> est une classe d'aide utilisée dans l'implémentation de la classe CRedBlackTree<T>.

Description

La classe CRedBlackTreeNode<T> représente un noeud de CRedBlackTree<T>. Les méthodes de navigation dans l'arbre sont implémentées dans la classe.

Déclaration

```
template<typename T>
class CRedBlackTreeNode
```

Fichier header

```
#include <Generic\RedBlackTree.mqh>
```

Méthodes de classe

Méthode	Description
Value	Retourne et définit la valeur d'un noeud
Parent	Retourne et définit un pointeur vers le noeud parent
Left	Retourne et définit un pointeur vers le noeud à gauche
Right	Retourne et définit un pointeur vers le noeud à droite
Color	Retourne et définit la couleur d'un noeud
IsLeaf	Détermine si le noeud spécifié est une feuille
CreateEmptyNode	Crée un nouveau noeud noir sans parent ni enfant, et retourne un pointeur sur ce noeud

© 2000-2024, MetaQuotes Ltd.

Value (méthode Get)

Retourne la valeur du noeud.

```
T Value();
```

Valeur de Retour

Retourne la valeur du noeud.

Value (méthode Set)

Définit la valeur du noeud.

```
void Value(
   T  value     // valeur du noeud
   );
```

Paramètres

value

[in] Valeur du noeud.

© 2000-2024, MetaQuotes Ltd.

Parent (méthode Get)

Retourne un pointeur vers le noeud parent.

```
CRedBlackTreeNode<T>* Parent();
```

Valeur de Retour

Retourne un pointeur vers le noeud parent.

Parent (méthode Set)

Définit un pointeur vers le noeud parent.

```
void Parent(
   CRedBlackTreeNode<T>*  node     // un pointeur vers le noeud parent
   );
```

Paramètres

node

[in] Un pointeur vers le noeud parent.

© 2000-2024, MetaQuotes Ltd.

Left (méthode Get)

Retourne un pointeur vers le noeud à gauche.

```
CRedBlackTreeNode<T>* Left();
```

Valeur de Retour

Retourne un pointeur vers le noeud à gauche.

Left (méthode Set)

Définit un pointeur vers le noeud à gauche.

```
void Left(
   CRedBlackTreeNode<T>*  node     // un pointeur vers le noeud à gauche
   );
```

Paramètres

node

[in] Un pointeur vers le noeud à gauche.

© 2000-2024, MetaQuotes Ltd.

Right (méthode Get)

Retourne un pointeur vers le noeud à droite.

```
CRedBlackTreeNode<T>* Right();
```

Valeur de Retour

Retourne un pointeur vers le noeud à droite.

Right (méthode Set)

Définit un pointeur vers le noeud à droite.

```
void Right(
   CRedBlackTreeNode<T>*  node     // un pointeur vers le noeud à droite
   );
```

Paramètres

node

[in] Un pointeur vers le noeud à droite.

© 2000-2024, MetaQuotes Ltd.

Color (méthode Get)

Retourne la couleur d'un noeud.

```
ENUM_RED_BLACK_TREE_NODE_TYPE Color();
```

Valeur de Retour

Retourne la couleur d'un noeud.

Color (méthode Set)

Définit la couleur d'un noeud.

```
void Color(
   ENUM_RED_BLACK_TREE_NODE_TYPE  clr     // couleur du noeud
   );
```

Paramètres

clr

[in] Couleur du noeud.

Note

La couleur du noeud est définie en utilisant une valeur de l'énumération ENUM_RED_BLACK_TREE_NODE_TYPE. Elle peut être de 2 types :

- RED_BLACK_TREE_NODE_RED — couleur rouge du noeud ;
- RED_BLACK_TREE_NODE_BLACK — couleur noir du noeud.

© 2000-2024, MetaQuotes Ltd.

IsLeaf

Détermine si le noeud spécifié est une feuille.

```
bool IsLeaf();
```

Valeur de Retour

Retourne true si le noeud est une feuille, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

CreateEmptyNode

Crée un nouveau noeud noir sans parent ni enfant, et retourne un pointeur sur ce noeud.

```
static CRedBlackTreeNode<T>* CreateEmptyNode();
```

Valeur de Retour

Retourne un pointeur sur le nouveau noeud.

© 2000-2024, MetaQuotes Ltd.

CLinkedListNode<T>

CLinkedListNode<T> est une classe d'aide utilisée dans l'implémentation de la classe CLinkedListNode<T>.

Description

La classe CLinkedListNode<T> représente un noeud d'une liste doublement chaînée CLinkedListNode<T>. Les méthodes de navigation dans la liste sont implémentées dans la classe.

Déclaration

```
template<typename T>
class CLinkedListNode
```

Fichier header

```
#include <Generic\LinkedList.mqh>
```

Méthodes de classe

Méthode	Description
List	Retourne et définit un pointeur vers la CLinkedList<T>
Next	Retourne et définit un pointeur vers le noeud suivant
Previous	Retourne et définit un pointeur vers le noeud précédent
Value	Retourne et définit la valeur du noeud

© 2000-2024, MetaQuotes Ltd.

List (méthode Get)

Retourne un pointeur vers la CLinkedList<T>.

```
CLinkedList<T>* List();
```

Valeur de Retour

Retourne un pointeur vers la liste chaînée CLinkedList<T>.

List (méthode Set)

Définit un pointeur vers la CLinkedList<T>.

```
void List(
   CLinkedList<T>*  value     // un pointeur vers la liste
   );
```

Paramètres

value

[in] Un pointeur vers la liste chaînée CLinkedList<T>.

© 2000-2024, MetaQuotes Ltd.

Next (méthode Get)

Retourne un pointeur vers le noeud suivant.

```
CLinkedListNode<T>* Next();
```

Valeur de Retour

Retourne un pointeur vers le noeud suivant.

Next (méthode Set)

Définit un pointeur vers le noeud suivant.

```
void Next(
   CLinkedListNode<T>*  value     // un pointeur vers le noeud suivant
   );
```

Paramètres

value

[in] Un pointeur vers le noeud suivant.

© 2000-2024, MetaQuotes Ltd.

Previous (méthode Get)

Retourne un pointeur vers le noeud précédent.

```
CLinkedListNode<T>* Previous();
```

Valeur de Retour

Retourne un pointeur vers le noeud précédent.

Previous (méthode Set)

Définit un pointeur vers le noeud précédent.

```
void Previous(
   CLinkedListNode<T>*  value     // un pointeur vers le noeud précédent
   );
```

Paramètres

value

[in] Un pointeur vers le noeud précédent.

© 2000-2024, MetaQuotes Ltd.

Value (méthode Get)

Retourne la valeur du noeud.

```
T Value();
```

Valeur de Retour

Retourne la valeur du noeud.

Value (méthode Set)

Définit la valeur du noeud.

```
void Value(
   T  value     // Valeur du noeud
   );
```

Paramètres

value

[in] Valeur du noeud.

© 2000-2024, MetaQuotes Ltd.

CKeyValuePair<TKey, TValue>

La classe CKeyValuePair<TKey,TValue> implémente une paire clé/valeur.

Description

La classe CKeyValuePair<TKey,TValue> implémente lees méthodes permettant de travailler avec la clé et la valeur d'une paire clé/valeur.

Déclaration

```
template<typename TKey, typename TValue>
class CKeyValuePair : public IComparable<CKeyValuePair<TKey,TValue>*>
```

Fichier header

```
#include <Generic\HashMap.mqh>
```

Hiérarchie d'Héritage

IEqualityComparable
  IComparable
    CKeyValuePair

Méthodes de classe

Méthode	Description
Key	Retourne et définit la clé d'une paire clé/valeur
Value	Retourne et définit la valeur d'une paire clé/valeur
Clone	Crée une nouvelle paire clé/valeur dont la clé et la valeur sont égales aux données courante
Compare	Compare la paire clé/valeur actuelle avec celle spécifiée
Equals	Vérifie si la paire clé/valeur actuelle est égale à la paire spécifiée
HashCode	Calcule la valeur de hashage basée sur la paire clé/valeur

© 2000-2024, MetaQuotes Ltd.

Key (méthode Get)

Retourne la clé d'une paire clé/valeur.

```
TKey Key();
```

Valeur de Retour

Retourne la clé.

Key (méthode Set)

Définit la clé d'une paire clé/valeur.

```
void Key(
   TKey  key     // clé
   );
```

Paramètres

key

[in] Clé.

© 2000-2024, MetaQuotes Ltd.

Value (méthode Get)

Retourne la valeur d'une paire clé/valeur.

```
TValue Value();
```

Valeur de Retour

Retourne la valeur.

Value (méthode Set)

Définit la valeur d'une paire clé/valeur.

```
void Value(
   TValue  value     // valeur
   );
```

Paramètres

value

[in] Valeur

© 2000-2024, MetaQuotes Ltd.

Clone

Crée une nouvelle paire clé/valeur dont la clé et la valeur sont égales aux données courante.

```
TValue>* Clone();
```

Valeur de Retour

Retourne une nouvelle paire clé/valeur

© 2000-2024, MetaQuotes Ltd.

Compare

Compare la paire clé/valeur actuelle avec celle spécifiée.

```
int Compare(
   CKeyValuePair<TKeyTValue>*  pair     // la paire à comparer
   );
```

Paramètres

pair

[in] La paire à comparer.

Valeur de Retour

Retourne un nombre exprimant le rapport des paires clé/valeur actuelle et spécifiée :

- si le résultat est inférieur à 0, la paire clé/valeur actuelle est inférieure à celle passée en argument
- si le résultat est égal à 0, les deux paires clé/valeur sont égales
- si le résultat est supérieur à 0, la paire clé/valeur actuelle est supérieure à celle passée en argument

Note

La comparaison des paires clé/valeur s'effectue sur leurs clés.

© 2000-2024, MetaQuotes Ltd.

Equals

Vérifie si la paire clé/valeur actuelle est égale à la paire spécifiée.

```
bool Equals(
   CKeyValuePair<TKeyTValue>*  pair     // la paire à comparer
   );
```

Paramètres

pair

[in] La paire à comparer.

Valeur de Retour

Retourne true si les paires clé/valeur sont égales, false sinon.

Note

La comparaison des paires clé/valeur s'effectue sur leurs clés.

© 2000-2024, MetaQuotes Ltd.

HashCode

Calcule la valeur de hashage basée sur la paire clé/valeur.

```
int HashCode();
```

Valeur de Retour

Retourne le code de hashage.

Note

Le code de hashage de la paire clé/valeur est égal au code de hashage de la clé.

© 2000-2024, MetaQuotes Ltd.

CArrayList<T>

CArrayList<T> est une classe générique qui implémente l'interface IList<T>.

Description

La classe CArrayList<T> est une implémentation de la liste dynamique de données de type T. Cette classe fournit les méthodes de base pour travailler avec la liste, tel que l'accès à un élément par son indice, la recherche et la suppression d'éléments, le tri, etc.

Déclaration

```
template<typename T>
class CArrayList : public IList<T>
```

Fichier header

```
#include <Generic\ArrayList.mqh>
```

Hiérarchie d'Héritage

ICollection
  IList
    CArrayList

Méthodes de classe

Méthode	Description
Capacity	Retourne et définit la capacité actuelle d'une liste
Count	Retourne le nombre d'éléments dans la liste
Contains	Détermine si une liste contient un élément ayant la valeur spécifiée
TrimExcess	Définit la capacité d'une liste au nombre actuel d'éléments
TryGetValue	Retourne l'élément de la liste situé à l'indice spécifié
TrySetValue	Définit la valeur de l'élément de la liste situé à l'indice spécifié
Add	Ajoute un élément dans la liste
AddRange	Ajoute une collection ou un tableau d'éléments à la liste
Insert	Insère un élément dans la liste à l'indice spécifié
InsertRange	Insère une collection ou un tableau d'éléments dans la liste à l'indice spécifié
CopyTo	Copie tous les éléments d'une liste dans le tableau spécifié en commençant à l'indice spécifié
BinarySearch	Recherche la valeur spécifiée dans une liste triée en ordre ascendant
IndexOf	Recherche la première occurrence d'une valeur dans une liste

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
LastIndexOf	Recherche la dernière occurrence d'une valeur dans une liste
Clear	Supprime tous les éléments d'une collection
Remove	Supprime la première occurrence de l'élément spécifié dans la liste
RemoveAt	Supprime l'élément situé à l'indice spécifié dans la liste
RemoveRange	Supprime un intervalle d'éléments de la liste
Reverse	Inverse l'ordre des éléments dans la liste
Sort	Trie les éléments de la liste

© 2000-2024, MetaQuotes Ltd.

Capacity (méthode Get)

Retourne la capacité actuelle de la liste.

```
int Capacity();
```

Valeur de Retour

Retourne la capacité actuelle de la liste.

Capacity (méthode Set)

Définit la capacité actuelle d'une liste.

```
void Capacity(
   const int  capacity     // capacité
   );
```

Paramètres

capacity

[in] Une nouvelle valeur pour la capacité.

© 2000-2024, MetaQuotes Ltd.

Count

Retourne le nombre d'éléments dans la liste.

```
int Count();
```

Valeur de Retour

Retourne le nombre d'éléments.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si la liste contient un élément ayant la valeur spécifiée.

```
bool Contains(
   T  item     // la valeur à chercher
   );
```

Paramètres

item

[in] La valeur à chercher.

Valeur de Retour

Retourne true si un élément ayant la valeur spécifié a été trouvé dans la liste, false sinon.

© 2000-2024, MetaQuotes Ltd.

TrimExcess

Définit la capacité d'une liste au nombre actuel d'éléments, libérant ainsi la mémoire non utilisée.

```
void TrimExcess();
```

© 2000-2024, MetaQuotes Ltd.

TryGetValue

Retourne l'élément de la liste situé à l'indice spécifié.

```
bool TryGetValue(
   const int  index,      // indice
   T&         value       // variable reçevant l'élément trouvé
   );
```

Paramètres

index

[in] L'indice de l'élément de la liste désiré.

&value

[out] La variable reçevant l'élément.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

TrySetValue

Définit la valeur de l'élément de la liste situé à l'indice spécifié.

```
bool TrySetValue(
   const int  index,      // indice
   T          value       // valeur de l'élément
   );
```

Paramètres

index

[in] L'indice de l'élément de la liste à modifier.

value

[in] Définit la valeur de l'élément de la liste.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément dans la liste.

```
bool Add(
   T  value     // la valeur de l'élément
   );
```

Paramètres

value

[in] La valeur de l'élément à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

AddRange

Ajoute une collection ou un tableau d'éléments à la liste.

Version ajoutant un tableau.

```
bool AddRange(
   const T&  array[]               // tableau devant être ajouté
   );
```

Version ajoutant une collection.

```
bool AddRange(
   ICollection<T>*  collection     // collection devant être ajoutée
   );
```

Paramètres

&array[]

[in] Un tableau devant être ajouté.

**collection*

[in] Une collection devant être ajoutée.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Insert

Insère un élément dans la liste à l'indice spécifié.

```
bool Insert(
   const int  index,      // indice où insérer
   T          item        // la valeur à insérer
   );
```

Paramètres

index

[in] l'indiceoù la valeur sera insérée.

item

[in] La valeur à insérer à l'indice spécifié.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

InsertRange

Insère une collection ou un tableau d'éléments dans la liste à l'indice spécifié.

Version insérant un tableau.

```
bool InsertRange(
   const int  index,               // indice où insérer
   const T&   array[]              // le tableau à insérer
   );
```

Version insérant une collection.

```
bool InsertRange(
   const int        index,         // indice où insérer
   ICollection<T>*  collection     // la collection à insérer
   );
```

Paramètres

index

[in] l'indiceoù la valeur sera insérée.

&array[]

[in] Un tableau à insérer à l'indice spécifié.

**collection*

[in] Une collection à insérer à l'indice spécifié.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie tous les éléments d'une liste dans le tableau spécifié en commençant à l'indice spécifié.

```
int CopyTo(
   T&          dst_array[],      // le tableau de destination
   const int   dst_start=0       // l'indice de départ for writing
   );
```

Paramètres

&dst_array[]

[out] Un tableau dans lequel les éléments de la liste seront écrits.

dst_start=0

[in] Un indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre d'éléments copiés.

© 2000-2024, MetaQuotes Ltd.

BinarySearch

Recherche la valeur spécifiée dans une liste triée en ordre ascendant.

Version cherchant dans l'intervalle spécifié de valeurs en utilisant la classe qui implémente l'interface IComparable<T> pour comparer les éléments.

```
int BinarySearch(
   const int      index,        // l'indice de départ
   const int      count,        // l'intervalle de recherche
   T              item,         // la valeur à chercher
   IComparer<T>*  comparer      // interface vers le comparateur
   );
```

Version utilisant la classe qui implémente l'interface IComparable<T> pour comparer les éléments.

```
int BinarySearch(
   T              item,         // la valeur à chercher
   IComparer<T>*  comparer      // interface vers le comparateur
   );
```

Version utilisant la méthode globale ::Compare pour comparer les éléments.

```
int BinarySearch(
   T  item                      // la valeur à chercher
   );
```

Paramètres

index

[in] L'indice de départ à partir duquel la recherche commence.

count

[in] La longueur de l'intervalle de recherche.

item

[in] La valeur à chercher.

**comparer*

[in] Une interface pour comparer les éléments.

Valeur de Retour

Retourne l'indice de l'élément trouvé. Si la valeur recherchée n'est pas trouvée, retourne l'indice de l'élément le plus petit qui est le plus proche de la valeur.

© 2000-2024, MetaQuotes Ltd.

IndexOf

Recherche la première occurrence d'une valeur dans une liste.

Version cherchant dans la liste entière.

```
int IndexOf(
   T  item                    // la valeur à chercher
   );
```

Version cherchant à partir de la position spécifiée et jusqu'à la fin de la liste.

```
int IndexOf(
   T          item,           // la valeur à chercher
   const int  start_index     // l'indice de départ
   );
```

Version cherchant à partir de la position spécifiée dans l'intervalle spécifié.

```
int IndexOf(
   T          item,           // la valeur à chercher
   const int  start_index,    // l'indice de départ
   const int  count           // l'intervalle de recherche
   );
```

Paramètres

item

[in] La valeur à chercher.

start_index

[in] L'indice de départ à partir duquel la recherche commence.

count

[in] La longueur de l'intervalle de recherche.

Valeur de Retour

Retourne l'indice du premier élément trouvé. Si la valeur n'est pas trouvée, retourne -1.

© 2000-2024, MetaQuotes Ltd.

LastIndexOf

Recherche la dernière occurrence d'une valeur dans une liste.

Version cherchant dans la liste entière.

```
int LastIndexOf(
   T  item                    // la valeur à chercher
   );
```

Version cherchant à partir de la position spécifiée et jusqu'à la fin de la liste.

```
int LastIndexOf(
   T          item,           // la valeur à chercher
   const int  start_index     // l'indice de départ
   );
```

Version cherchant à partir de la position spécifiée dans l'intervalle spécifié.

```
int LastIndexOf(
   T          item,           // la valeur à chercher
   const int  start_index,    // l'indice de départ
   const int  count           // l'intervalle de recherche
   );
```

Paramètres

item

[in] La valeur à chercher.

start_index

[in] L'indice de départ à partir duquel la recherche commence.

count

[in] La longueur de l'intervalle de recherche.

Valeur de Retour

Retourne l'indice du dernier élément trouvé. Si la valeur n'est pas trouvée, retourne -1.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments d'une collection.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime la première occurrence de l'élément spécifié dans la liste.

```
bool Remove(
   T  item     // l'élément par valeur
   );
```

Paramètres

item

[in] La valeur de l'élément à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

RemoveAt

Supprime l'élément situé à l'indice spécifié dans la liste.

```
bool RemoveAt(
   const int  index     // indice
   );
```

Paramètres

index

[in] L'indice de l'élément à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

RemoveRange

Supprime un intervalle d'éléments de la liste.

```
bool RemoveRange(
   const int  start_index,      // l'indice de départ
   const int  count             // le nombre d'éléments
   );
```

Paramètres

start_index

[in] L'indice de départ à partir duquel la suppression commence.

count

[in] Le nombre d'éléments à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Reverse

Inverse l'ordre des éléments dans la liste.

Version opérant sur la liste entière.

```
bool Reverse();
```

Version opérant sur l'intervalle spécifié d'éléments de la liste.

```
bool Reverse(
   const int  start_index,     // l'indice de départ
   const int  count            // le nombre d'éléments
   );
```

Paramètres

start_index

[in] L'indice de départ.

count

[in] Le nombre d'éléments de la liste participant à l'opération.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Sort

Trie les éléments de la liste.

Version triant tous les éléments de la liste.

```
bool Sort();
```

Version triant tous les éléments de la liste en utilisant une classe qui implémente l'interface IComparable<T> pour comparer les éléments.

```
bool Sort(
   IComparer<T>*  comparer         // interface pour comparer
   );
```

Version triant l'intervalle spécifié d'éléments dans la liste en utilisant une classe qui implémente l'interface IComparable<T> pour comparer les éléments.

```
bool Sort(
   const int  start_index,     // l'indice de départ
   const int  count            // le nombre d'éléments
   IComparer<T>*  comparer     // interface vers le comparateur
   );
```

Paramètres

**comparer*

[in] Une interface pour comparer les éléments.

start_index

[in] L'indice de départ à partir duquel le tri commence.

count

[in] La longueur de l'intervalle de tri.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

CHashMap<TKey, TValue>

CHashMap<TKey, TValue> est une classe générique qui implémente l'interface IMap<TKey, TValue>

Description

La classe CHashMap<TKey, TValue> est une implémentation de la table dynamique de hashage, dont les données sont stockées sous la forme de paires clé/valeur non triées et prenant en compte le besoin d'unicité de la clé. Cette classe fournit les méthodes de base pour travailler avec une table de hashage, tel que l'accès à une valeur par sa clé, la recherche et la suppression du paire clé/valeur, etc.

Déclaration

```
template<typename TKey, typename TValue>
class CHashMap : public IMap<TKey, TValue>
```

Fichier header

```
#include <Generic\HashMap.mqh>
```

Hiérarchie d'Héritage

ICollection
  IMap
    CHashMap

Méthodes de classe

Méthode	Description
Add	Ajoute une paire clé/valeur dans la table de hashage
Count	Retourne le nombre d'éléments dans la table de hashage.
Comparer	Retourne un pointeur sur l'interface IEqualityComparer<T>, utilisé pour organiser une table de hashage.
Contains	Détermine si la table de hashage contient la paire clé/valeur spécifiée
ContainsKey	Détermine si la table de hashage contient la paire clé/valeur avec la clé spécifiée
ContainsValue	CHashMap<TKey, TValue> est une classe générique qui implémente l'interface IMap<TKey, TValue>
CopyTo	Copie toutes les paires clé/valeur de la table de hashage vers les tableaux spécifiés, en commençant à l'indice spécifié
Clear	Supprime tous les éléments de la table de hashage
Remove	Supprime la première occurrence de la paire clé/valeur de la table de hashage
TryGetValue	Retourne l'élément ayant la clé spécifiée dans la table de hashage

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
TrySetValue	Change la valeur d'une paire clé/valeur de la table de hashage ayant la clé spécifiée

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute une paire clé/valeur dans la table de hashage.

Version ajouter une paire clé/valeur générée.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // la paire clé/valeur
   );
```

Version ajoutant une nouvelle paire clé/valeur avec les clé et valeur spécifiées.

```
bool Add(
   TKey    key,                         // clé
   TValue  value                        // valeur
   );
```

Paramètres

pair

[in] La pair clé/valeur.

key

[in] Clé.

value

[in] Valeur.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Count

Retourne le nombre d'éléments dans la table de hashage.

```
int Count();
```

© 2000-2024, MetaQuotes Ltd.

Comparer

Retourne un pointeur sur l'interface IEqualityComparer<T>, utilisé pour organiser une table de hashage.

```
IEqualityComparer<TKey>* Comparer() const;
```

Valeur de Retour

Retourne un pointeur sur l'interface IEqualityComparer<T>.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si la table de hashage contient la paire clé/valeur spécifiée.

Version utilisant une paire clé/valeur générée.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // la paire clé/valeur
   );
```

Version utilisant une clé et une valeur distinctes.

```
bool Contains(
   TKey   key,                          // clé
   TValue value                         // valeur
   );
```

Paramètres

**item*

[in] La pair clé/valeur.

key

[in] Clé.

value

[in] Valeur.

Valeur de Retour

Retourne true si la table de hashage contient la paire clé/valeur avec les clé et valeur spécifiées, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

ContainsKey

Détermine si la table de hashage contient la paire clé/valeur avec la clé spécifiée.

```
bool ContainsKey(
   TKey  key     // clé
   );
```

Paramètres

key

[in] Clé.

Valeur de Retour

Retourne true si la table de hashage contient la paire clé/valeur avec la clé spécifiée, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

ContainsValue

Détermine si la table de hashage contient la paire clé/valeur avec la valeur spécifiée.

```
bool ContainsValue(
   TValue  value     // valeur
   );
```

Paramètres

value

[in] Valeur.

Valeur de Retour

Retourne true si la table de hashage contient la paire clé/valeur avec la valeur spécifiée, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie toutes les paires clé/valeur de la table de hashage vers les tableaux spécifiés, en commençant à l'indice spécifié.

Version copiant une table de hashage vers le tableau de paires clé/valeur.

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // le tableau de destination key/valu
   const int                    dst_start=0      // l'indice de départ for writing
   );
```

Version copiant une table de hashage vers des tableaux de paires clé/valeur séparés.

```
int CopyTo(
   TKey&      dst_keys[],                        // le tableau de destination keys
   TValue&    dst_values[],                      // le tableau de destination values
   const int  dst_start=0                        // indice de début d'écriture
   );
```

Paramètres

&dst_array[]

[out] Un tableau vers lequel toutes les paires de la table de hashage seront écrites.

&dst_keys[]

[out] Un tableau vers lequel toutes les clés de la table de hashage seront écrites.

&dst_values[]

[out] Un tableau vers lequel toutes les valeurs de la table de hashage seront écrites.

dst_start=0

[in] L'indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre de paires clé/valeurs copiées.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments de la table de hashage.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime la première occurrence de la paire clé/valeur de la table de hashage.

Version supprimant une paire clé/valeur basée sur la paire clé/valeur générée.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // la paire clé/valeur"
   );
```

Version supprimant une paire clé/valeur basée sur la clé.

```
bool Remove(
   TKey  key                            // clé
   );
```

Paramètres

**item*

[in] La pair clé/valeur.

key

[in] Clé.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

TryGetValue

Retourne l'élément de la table de hashage ayant la clé spécifiée.

```
bool TryGetValue(
   TKey     key,       // clé
   TValue&  value      // une variable pour écrire la valeur
   );
```

Paramètres

key

[in] Clé.

&value

[out] La variable dans laquelle la valeur spécifiée de la paire clé/valeur sera écrite.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

TrySetValue

Change la valeur de la paire clé/valeur de la table de hashage ayant la clé spécifiée.

```
bool TrySetValue(
   TKey    key,       // clé
   TValue  value      // nouvelle valeur
   );
```

Paramètres

key

[in] Clé.

value

[in] La nouvelle valeur à assigner à la paire clé/valeur spécifiée.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

CHashSet<T>

CHashSet<T> est une classe générique qui implémente l'interface ISet<T>.

Description

La classe CHashSet<T> est une implémentation de l'ensemble (set) dynamique et non ordonné de données de type T, avec l'unicité de chaque valeur. Cette classe fournit les méthodes de base pour travailler avec les sets et toutes les opérations liées, telles que : l'union et l'intersection des sets, la définition de sous-ensembles stricts et non stricts, etc.

Déclaration

```
template<typename T>
class CHashSet : public ISet<T>
```

Fichier header

```
#include <Generic\HashSet.mqh>
```

Hiérarchie d'Héritage

ICollection
  ISet
    CHashSet

Méthodes de classe

Méthode	Description
Add	Ajoute un élément dans un set
Count	Retourne le nombre d'éléments d'un set
Comparer	Détermine si un set contient un élément ayant la valeur spécifiée
Contains	Retourne un pointeur sur l'interface IEqualityComparer<T>, utilisé pour organiser un set
TrimExcess	Définit la capacité d'un set au nombre actuel d'éléments, libérant ainsi la mémoire non utilisée.
CopyTo	Copie tous les éléments d'un set dans le tableau spécifié en commençant à l'indice spécifié
Clear	Supprime tous les éléments d'un set
Remove	Supprime l'élément spécifié d'un set
ExceptWith	Effectue l'opération de différence entre la collection courante et la collection passée en argument (tableau)
IntersectWith	Effectue l'opération d'intersection entre la collection courante et la collection passée en argument (tableau)

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
SymmetricExceptWith	Effectue l'opération de différence symétrique entre la collection courante et la collection passée en argument (tableau)
UnionWith	Effectue l'union entre la collection courante et la collection passée en argument (tableau)
IsProperSubsetOf	Détermine si le set actuel est un vrai sous-ensemble de la collection ou du tableau spécifié
IsProperSupersetOf	Détermine si le set actuel est un vrai sur-ensemble de la collection ou du tableau spécifié
IsSubsetOf	Détermine si le set actuel est un sous-ensemble de la collection ou du tableau spécifié
IsSupersetOf	Détermine si le set actuel est un sur-ensemble de la collection ou du tableau spécifié
Overlaps	Détermine si le set actuel chevauche la collection ou le tableau spécifié
SetEquals	Détermine si le set actuel contient tous les éléments de la collection ou du tableau spécifié

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément dans un set.

```
bool Add(
   T  value     // la valeur de l'élément
   );
```

Paramètres

value

[in] La valeur de l'élément à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Count

Retourne le nombre d'éléments d'un set.

```
int Count();
```

Valeur de Retour

Retourne le nombre d'éléments.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si un set contient un élément ayant la valeur spécifiée.

```
bool Contains(
   T  item     // la valeur à chercher
   );
```

Paramètres

item

[in] La valeur à chercher.

Valeur de Retour

Retourne true si un élément ayant la valeur spécifié a été trouvé dans le set, false sinon.

© 2000-2024, MetaQuotes Ltd.

Comparer

Retourne un pointeur sur l'interface IEqualityComparer<T>, utilisé pour organiser un set.

```
IEqualityComparer<T>* Comparer() const;
```

Valeur de Retour

Retourne un pointeur sur l'interface IEqualityComparer<T>.

© 2000-2024, MetaQuotes Ltd.

TrimExcess

Définit la capacité d'un set au nombre actuel d'éléments, libérant ainsi la mémoire non utilisée.

```
void TrimExcess();
```

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie tous les éléments d'un set dans le tableau spécifié en commençant à l'indicespécifié.

```
int CopyTo(
   T&          dst_array[],      // le tableau de destination
   const int   dst_start=0       // indice de début d'écriture
   );
```

Paramètres

&dst_array[]

[out] Un tableau dans lequel les éléments du set seront écrits.

dst_start=0

[in] Un indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre d'éléments copiés.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments d'un set.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime l'élément spécifié d'un set.

```
bool Remove(
   T  item     // l'élément par valeur
   );
```

Paramètres

item

[in] La valeur de l'élément à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

ExceptWith

Effectue l'opération de différence entre la collection courante et la collection passée en argument (tableau). Elle enlève de la collection courante (tableau) tous les éléments qui se trouvent dans la collection spécifiée (tableau).

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void ExceptWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection à sortir du set actuel.

&collection[]

[in] Un tableau à sortir du set actuel.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

IntersectWith

Effectue l'opération d'intersection entre la collection courante et la collection passée en argument (tableau). Elle modifie la collection courante pour ne contenir que les éléments qui se trouvent dans la collection spécifiée (tableau).

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void IntersectWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection avec laquelle opérer l'intersection avec le set actuel.

&collection[]

[in] Un tableau avec lequel opérer l'intersection avec le set actuel.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

SymmetricExceptWith

Effectue l'opération de différence symétrique entre la collection courante et la collection passée en argument (tableau). Elle modifie la collection courante pour ne contenir que les éléments qui se trouvent dans l'objet source ou dans la collection spécifiée (tableau) mais pas dans les deux.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void SymmetricExceptWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection avec laquelle produire la différence symétrique.

&collection[]

[in] Un tableau avec lequel produire la différence symétrique.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

UnionWith

Effectue l'union entre la collection courante et la collection passée en argument (tableau). Elle ajoute les éléments manquants de la collection spécifiée (tableau) dans la collection courante (array).

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void UnionWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection avec laquelle opérer l'union avec le set actuel.

&collection[]

[in] Un tableau avec lequel opérer l'union avec le set actuel.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

IsProperSubsetOf

Détermine si le set actuel est un vrai sous-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsProperSubsetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set courant est un vrai sous-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

IsProperSupersetOf

Détermine si le set actuel est un vrai sur-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsProperSupersetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set courant est un vrai sur-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

IsSubsetOf

Détermine si le set actuel est un sous-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsSubsetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set courant est un sous-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

IsSupersetOf

Détermine si le set actuel est un sur-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsSupersetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set courant est un sur-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

Overlaps

Détermine si le set actuel chevauche la collection ou le tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // une collection à comparer
   );
```

Version utilisant un tableau.

```
bool Overlaps(
   T&  array[]                     // un tableau à comparer
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer le chevauchement.

&collection[]

[in] Un tableau pour déterminer le chevauchement.

Valeur de Retour

Retourne true si le set actuel et une collection ou un tableau se chevauchent, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

SetEquals

Détermine si le set actuel contient tous les éléments de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // une collection à comparer
   );
```

Version utilisant un tableau.

```
bool SetEquals(
   T&  array[]                     // un tableau à comparer
   );
```

Paramètres

**collection*

[in] Une collection d'éléments à comparer.

&collection[]

[in] Un tableau d'éléments à comparer.

Valeur de Retour

Retourne true si le set actuel contient tous les éléments de la collection ou du tableau spécifié, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

CLinkedList<T>

CLinkedList<T> est une classe générique qui implémente l'interface ICollection<T>.

Description

La classe CLinkedList<T> est une implémentation de la liste dynamique doublement chaînée de données de type T. Cette classe fournit les méthodes de base pour travailler avec les doublement chaînée, tel que l'ajout, la suppression, la recherche d'éléments, etc.

Déclaration

```
template<typename T>
class CLinkedList : public ICollection<T>
```

Fichier header

```
#include <Generic\LinkedList.mqh>
```

Hiérarchie d'Héritage

ICollection
  CLinkedList

Méthodes de classe

Méthode	Description
Add	Ajoute un élément dans une liste chaînée
AddAfter	Ajoute un élément dans la liste chaînée après le noeud spécifié
AddBefore	Ajoute un élément dans la liste chaînée avant le noeud spécifié
AddFirst	Ajoute un élément au début de la liste chaînée
AddLast	Ajoute un élément à la fin de la liste chaînée
Count	Retourne le nombre d'éléments dans la liste chaînée
Head	Retourne un pointeur sur le premier noeud de la liste chaînée
First	Retourne un pointeur sur le premier noeud de la liste chaînée
Last	Retourne un pointeur vers le dernier noeud de la liste chaînée.
Contains	Détermine si la liste chaînée contient un élément ayant la valeur spécifiée.
CopyTo	Copie tous les éléments de la liste chaînée dans le tableau spécifié en commençant à l'indice spécifié
Clear	Supprime tous les éléments d'une liste chaînée.
Remove	Supprime la première occurrence de l'élément spécifié dans la liste chaînée

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
RemoveFirst	Supprime le premier élément de la liste chaînée
RemoveLast	Supprime le dernier élément de la liste chaînée
Find	Recherche la première occurrence de la valeur spécifiée dans la liste chaînée
FindLast	Recherche la dernière occurrence de la valeur spécifiée dans la liste chaînée

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément dans une liste chaînée.

```
bool Add(
   T  value     // la valeur de l'élément
   );
```

Paramètres

value

[in] La valeur de l'élément à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

AddAfter

Ajoute un élément dans la liste chaînée après le noeud spécifié.

Version ajoutant un élément par valeur.

```
CLinkedListNode<T>* AddAfter(
   CLinkedListNode<T>*  node,         // le noeud après lequel l'élément doit être ajou
   T                    value         // l'élément à ajouter
   );
```

Valeur de Retour

Retourne un pointeur sur le noeud ajouté.

Version ajoutant un élément sous forme de noeud par valeur.

```
bool AddAfter(
   CLinkedListNode<T>*  node,         // le noeud après lequel l'élément doit être ajou
   CLinkedListNode<T>*  new_node      // le noeud à ajouter
   );
```

Paramètres

*node

[in] Le noeud de la liste chaînée après lequel le nouvel élément sera ajouté.

value

[in] Un élément à ajouter.

*new_node

[in] Un noeud à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

AddBefore

Ajoute un élément dans la liste chaînée avant le noeud spécifié

Version ajoutant un élément par valeur.

```
CLinkedListNode<T>* AddBefore(
   CLinkedListNode<T>*  node,         // le noeud avant lequel l'élément doit être ajou
   T                    value         // l'élément à ajouter
   );
```

Valeur de Retour

Retourne un pointeur sur le noeud ajouté.

Version ajoutant un élément sous forme de noeud par valeur.

```
bool AddBefore(
   CLinkedListNode<T>*  node,         // le noeud avant lequel l'élément doit être ajou
   CLinkedListNode<T>*  new_node      // le noeud à ajouter
   );
```

Paramètres

**node*

[in] Le noeud de la liste chaînée avant lequel le nouvel élément sera ajouté.

value

[in] Un élément à ajouter.

**new_node*

[in] Un noeud à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

AddFirst

Ajoute un élément au début de la liste chaînée.

Version ajoutant un élément par valeur.

```
CLinkedListNode<T>* AddFirst(
   T    value                      // un élément à ajouter
   );
```

Valeur de Retour

Retourne un pointeur sur le noeud ajouté.

Version ajoutant un élément sous forme de noeud par valeur.

```
bool AddFirst(
   CLinkedListNode<T>*  node      // le noeud à ajouter
   );
```

Paramètres

value

[in] Un élément à ajouter.

**node*

[in] Un noeud à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

AddLast

Ajoute un élément à la fin de la liste chaînée

Version ajoutant un élément par valeur.

```
CLinkedListNode<T>* AddLast(
   T  value                      // un élément à ajouter
   );
```

Valeur de Retour

Retourne un pointeur sur le noeud ajouté.

Version ajoutant un élément sous forme de noeud par valeur.

```
bool AddLast(
   CLinkedListNode<T>*  node     // le noeud à ajouter
   );
```

Paramètres

value

[in] Un élément à ajouter.

**node*

[in] Un noeud à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Count

Retourne le nombre d'éléments dans la liste chaînée.

```
int Count();
```

Valeur de Retour

Retourne le nombre d'éléments.

© 2000-2024, MetaQuotes Ltd.

Head

Retourne un pointeur vers le premier noeud de la liste chaînée.

```
CLinkedListNode<T>* Head();
```

Valeur de Retour

Retourne un pointeur sur le premier noeud.

© 2000-2024, MetaQuotes Ltd.

First

Retourne un pointeur vers le premier noeud de la liste chaînée.

```
CLinkedListNode<T>* First();
```

Valeur de Retour

Retourne un pointeur sur le premier noeud.

© 2000-2024, MetaQuotes Ltd.

Last

Retourne un pointeur vers le dernier noeud de la liste chaînée.

```
CLinkedListNode<T>* Last();
```

Valeur de Retour

Retourne un pointeur sur le dernier noeud.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si la liste chaînée contient un élément ayant la valeur spécifiée.

```
bool Contains(
   T  item     // la valeur à chercher
   );
```

Paramètres

item

[in] La valeur à chercher.

Valeur de Retour

Retourne true si un élément ayant la valeur spécifié a été trouvé dans la liste chaînée, false sinon.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie tous les éléments de la liste chaînée dans le tableau spécifié en commençant à l'indice spécifié.

```
int CopyTo(
   T&          dst_array[],      // le tableau de destination
   const int   dst_start=0       // l'indice de départ for writing
   );
```

Paramètres

&dst_array[]

[out] Un tableau dans lequel les éléments de la liste chaînée seront écrits.

dst_start=0

[in] Un indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre d'éléments copiés.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments d'une collection.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime la première occurrence de l'élément spécifié dans la liste chaînée.

Version supprimant un élément par valeur.

```
bool Remove(
   T  item                        // l'élément par valeur
   );
```

Version supprimant un élément par un pointeur vers un noeud.

```
bool Remove(
   CLinkedListNode<T>*  node      // le noeud élément
   );
```

Paramètres

item

[in] La valeur de l'élément à supprimer.

**node*

[in] Le noeud de l'élément à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

RemoveFirst

Supprime le premier élément de la liste chaînée.

```
bool RemoveFirst();
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

RemoveLast

Supprime le dernier élément de la liste chaînée.

```
bool RemoveLast();
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Find

Recherche la première occurrence de la valeur spécifiée dans la liste chaînée.

```
CLinkedListNode<T>* Find(
   T  value     // la valeur à chercher
   );
```

Paramètres

value

[in] La valeur à chercher.

Valeur de Retour

Retourne un pointeur vers le premier noeud contenant la valeur recherchée en cas de succès, sinon NULL.

© 2000-2024, MetaQuotes Ltd.

FindLast

Recherche la dernière occurrence de la valeur spécifiée dans la liste chaînée.

```
CLinkedListNode<T>* FindLast(
   T  value     // la valeur à chercher
   );
```

Paramètres

value

[in] La valeur à chercher.

Valeur de Retour

Retourne un pointeur vers le dernier noeud trouvé contenant la valeur recherchée en cas de succès, sinon NULL.

© 2000-2024, MetaQuotes Ltd.

CQueue<T>

CQueue<T> est une classe générique qui implémente l'interface ICollection<T>.

Description

La classe CQueue<T> est une collection dynamique de données de type T, organisée comme une queue opérant selon le principe FIFO (first in, first out).

Déclaration

```
template<typename T>
class CQueue : public ICollection<T>
```

Fichier header

```
#include <Generic\Queue.mqh>
```

Hiérarchie d'Héritage

ICollection
  CQueue

Méthodes de classe

Méthode	Description
Add	Ajoute un élément dans la queue
Enqueue	Ajoute un élément dans la queue
Count	Retourne le nombre d'éléments dans la queue
Contains	Détermine si la queue contient un élément ayant la valeur spécifiée
TrimExcess	Définit la capacité d'une queue au nombre actuel d'éléments, libérant ainsi la mémoire non utilisée
CopyTo	Copie tous les éléments d'une queue dans le tableau spécifié en commençant à l'indice spécifié
Clear	Supprime tous les éléments d'une queue
Remove	Supprime la première occurrence de l'élément spécifié dans la queue
Dequeue	Retourne le premier élément et l'enlève de la queue
Peek	Retourne le premier élément sans l'enlever de la queue

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément dans une queue.

```
bool Add(
   T  value     // la valeur de l'élément
   );
```

Paramètres

value

[in] La valeur de l'élément à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Enqueue

Ajoute un élément dans une queue.

```
bool Enqueue(
   T  value     // élément à ajouter
   );
```

Paramètres

value

[in] Un élément à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Count

Retourne le nombre d'éléments dans la queue.

```
int Count();
```

Valeur de Retour

Retourne le nombre d'éléments.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si la queue contient un élément ayant la valeur spécifiée.

```
bool Contains(
   T  item     // la valeur à chercher
   );
```

Paramètres

item

[in] La valeur à chercher.

Valeur de Retour

Retourne true si un élément ayant la valeur spécifié a été trouvé dans la queue, false sinon.

© 2000-2024, MetaQuotes Ltd.

TrimExcess

Définit la capacité d'une queue au nombre actuel d'éléments, libérant ainsi la mémoire non utilisée.

```
void TrimExcess();
```

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie tous les éléments d'une queue dans le tableau spécifié en commençant à l'indice spécifié.

```
int CopyTo(
   T&          dst_array[],      // le tableau de destination
   const int   dst_start=0       // l'indice de départ for writing
   );
```

Paramètres

&dst_array[]

[out] Un tableau dans lequel les éléments de la queue seront écrits.

dst_start=0

[in] Un indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre d'éléments copiés.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments d'une queue.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime la première occurrence de l'élément spécifié dans la queue.

```
bool Remove(
   T  item     // l'élément par valeur
   );
```

Paramètres

item

[in] La valeur de l'élément à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Dequeue

Retourne le premier élément et l'enlève de la queue.

```
T Dequeue();
```

Valeur de Retour

Retourne le premier élément.

© 2000-2024, MetaQuotes Ltd.

Peek

Retourne le premier élément sans l'enlever de la queue.

```
T Peek();
```

Valeur de Retour

Retourne le premier élément.

© 2000-2024, MetaQuotes Ltd.

CRedBlackTree<T>

CRedBlackTree<T> est une classe générique qui implémente l'interface ICollection<T>.

Description

La classe CRedBlackTree<T> est une implémentation de l'arbre dynamique rouge et noir dont les noeuds stockent des données de type T. La classe fournit les méthodes de base pour travailler avec les arbres bicolors, telles que l'ajout, la suppression, la recherche des valeurs minimum et maximum, etc.

Déclaration

```
template<typename T>
class CRedBlackTree : public ICollection<T>
```

Fichier header

```
#include <Generic\RedBlackTree.mqh>
```

Hiérarchie d'Héritage

ICollection
  CRedBlackTree

Méthodes de classe

Méthode	Description
Add	Ajoute un élément à un arbre rouge et noir
Root	Retourne un pointeur vers la racine de l'arbre rouge et noir
Count	Retourne le nombre d'éléments de l'arbre rouge et noir
Contains	Détermine si l'arbre rouge et noir contient un élément ayant la valeur spécifiée
Comparer	Retourne un pointeur vers l'interface IComparer<T> utilisé pour organiser un arbre rouge et noir
TryGetMin	Retourne l'élément minimum d'un arbre rouge et noir
TryGetMax	Retourne l'élément maximum d'un arbre rouge et noir
CopyTo	Copie tous les éléments d'un arbre rouge et noir dans le tableau spécifié en commençant à l'indice spécifié
Clear	Supprime tous les éléments d'un arbre rouge et noir
Remove	Supprime l'occurrence de l'élément spécifié dans un arbre rouge et noir
RemoveMin	Supprime un élément ayant la valeur minimum d'un arbre rouge et noir
RemoveMax	Supprime un élément ayant la valeur maximum d'un arbre rouge et noir
Find	Recherche l'occurence d'une valeur spécifiée dans un arbre rouge et noir

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
FindMax	Recherche un élément ayant la valeur maximum dans un arbre rouge et noir
FindMin	Recherche un élément ayant la valeur minimum dans un arbre rouge et noir

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément à un arbre rouge et noir.

```
bool Add(
   T  value     // élément à ajouter
   );
```

Paramètres

value

[in] Un élément à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Count

Retourne le nombre d'éléments de l'arbre rouge et noir.

```
int Count();
```

Valeur de Retour

Retourne le nombre d'éléments.

© 2000-2024, MetaQuotes Ltd.

Root

Retourne un pointeur vers la racine de l'arbre rouge et noir.

```
CRedBlackTreeNode<T>* Root();
```

Valeur de Retour

Retourne un pointeur vers la racine.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si l'arbre rouge et noir contient un élément ayant la valeur spécifiée.

```
bool Contains(
   T  item     // la valeur à chercher
   );
```

Paramètres

item

[in] La valeur à chercher.

Valeur de Retour

Retourne true si un élément ayant la valeur spécifiée est trouvé dans l'arbre rouge et noir, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

Comparer

Retourne un pointeur vers l'interface IComparer<T> utilisé pour organiser un arbre rouge et noir.

```
IComparer<T>* Comparer() const;
```

Valeur de Retour

Retourne un pointeur sur l'interface IComparer<T>.

© 2000-2024, MetaQuotes Ltd.

TryGetMin

Retourne l'élément minimum d'un arbre rouge et noir.

```
bool TryGetMin(
   T&  min     // une variable pour écrire la valeur
   );
```

Paramètres

&min

[out] La variable dans laquelle la valeur minimum sera écrite.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

TryGetMax

Retourne l'élément maximum d'un arbre rouge et noir.

```
bool TryGetMax(
   T&  max     // une variable pour écrire la valeur
   );
```

Paramètres

&max

[out] La variable dans laquelle la valeur maximum sera écrite.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie tous les éléments d'un arbre rouge et noir dans le tableau spécifié en commençant à l'indice spécifié.

```
int CopyTo(
   T&          dst_array[],      // le tableau de destination
   const int   dst_start=0       // indice de début d'écriture
   );
```

Paramètres

&dst_array[]

[out] Un tableau dans lequel les éléments de l'arbre rouge et noir seront écrits.

dst_start=0

[in] Un indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre d'éléments copiés.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments d'un arbre rouge et noir.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime l'occurrence de l'élément spécifié dans un arbre rouge et noir.

Version supprimant un élément avec la valeur spécifiée.

```
bool Remove(
   T  value                        // l'élément par valeur
   );
```

Version supprimant un élément par un pointeur vers un noeud.

```
bool Remove(
   CRedBlackTreeNode<T>*  node     // le noeud élément
   );
```

Paramètres

item

[in] La valeur de l'élément à supprimer.

**node*

[in] Le noeud de l'élément à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

RemoveMin

Supprime un élément ayant la valeur minimum d'un arbre rouge et noir.

```
bool RemoveMin();
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

RemoveMax

Supprime un élément ayant la valeur maximum d'un arbre rouge et noir.

```
bool RemoveMax();
```

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Find

Recherche l'occurence d'une valeur spécifiée dans un arbre rouge et noir.

```
CRedBlackTreeNode<T>* Find(
   T  value     // la valeur à chercher
   );
```

Paramètres

value

[in] La valeur à chercher.

Valeur de Retour

Retourne un pointeur vers le noeud trouvé contenant la valeur recherchée en cas de succès, sinon NULL.

© 2000-2024, MetaQuotes Ltd.

FindMin

Recherche un élément ayant la valeur minimum dans un arbre rouge et noir.

```
CRedBlackTreeNode<T>* FindMin();
```

Valeur de Retour

Retourne un pointeur vers le noeud trouvé contenant la valeur minimum en cas de succès, sinon NULL.

© 2000-2024, MetaQuotes Ltd.

FindMax

Recherche un élément ayant la valeur maximum dans un arbre rouge et noir.

```
CRedBlackTreeNode<T>* FindMax();
```

Valeur de Retour

Retourne un pointeur vers le noeud trouvé contenant la valeur maximum en cas de succès, sinon NULL.

© 2000-2024, MetaQuotes Ltd.

CSortedMap<TKey, TValue>

CSortedMap<TKey,TValue> est une classe générique qui implémente l'interface IMap<TKey,TValue>.

Description

La classe CSortedMap<TKey,TValue> est une implémentation d'une table de hashage dynamique dont les données sont stockées sous forme de paires clé/valeur, triées par la clé et prenant en compte le besoin d'unicité de la clé. Cette classe fournit les méthodes de base pour travailler avec une table de hashage, tel que l'accès à une valeur par sa clé, la recherche et la suppression du paire clé/valeur, etc.

Déclaration

```
template<typename TKey, typename TValue>
class CSortedMap : public IMap<TKey, TValue>
```

Fichier header

```
#include <Generic\SortedMap.mqh>
```

Hiérarchie d'Héritage

ICollection
  IMap
    CSortedMap

Méthodes de classe

Méthode	Description
Add	Ajoute une paire clé/valeur dans la table de hashage
Count	Retourne le nombre d'éléments dans la table de hashage triée.
Contains	Détermine si la table de hashage triée contient la paire clé/valeur spécifiée
ContainsKey	Détermine si la table de hashage triée contient la paire clé/valeur avec la clé spécifiée
ContainsValue	Détermine si la table de hashage triée contient la paire clé/valeur avec la valeur spécifiée
Comparer	Retourne un pointeur sur l'interface IComparer<T>, utilisé pour organiser une table de hashage triée.
CopyTo	Copie toutes les paires clé/valeur de la table de hashage triée vers les tableaux spécifiés, en commençant à l'indice spécifié
Clear	Supprime tous les éléments de la table de hashage triée
Remove	Supprime la première occurrence de la paire clé/valeur de la table de hashage triée

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
TryGetValue	Retourne l'élément ayant la clé spécifiée dans un set trié
TrySetValue	Change une paire clé/valeur avec la clé spécifiée dans la table de hashage triée

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute une paire clé/valeur dans la table de hashage.

Version ajouter une paire clé/valeur générée.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // la paire clé/valeur
   );
```

Version ajoutant une nouvelle paire clé/valeur avec les clé et valeur spécifiées.

```
bool Add(
   TKey    key,                         // clé
   TValue  value                        // valeur
   );
```

Paramètres

pair

[in] La pair clé/valeur.

key

[in] Clé.

value

[in] Valeur.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Count

Retourne le nombre d'éléments dans la table de hashage triée.

```
int Count();
```

© 2000-2024, MetaQuotes Ltd.

Comparer

Retourne un pointeur sur l'interface IComparer<T>, utilisé pour organiser une table de hashage triée.

```
IComparer<TKey>* Comparer() const;
```

Valeur de Retour

Retourne un pointeur sur l'interface IComparer<T>.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si la table de hashage triée contients la paire clé/valeur spécifiée.

Version utilisant une paire clé/valeur générée.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // la paire clé/valeur
   );
```

Version utilisant une clé et une valeur distinctes.

```
bool Contains(
   TKey    key,                         // clé
   TValue  value                        // valeur
   );
```

Paramètres

**item*

[in] La pair clé/valeur.

key

[in] Clé.

value

[in] Valeur.

Valeur de Retour

Retourne true si la table de hashage triée contient la pair clé/valeur ayant la clé et la valeur spécifiées, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

ContainsKey

Détermine si la table de hashage triée contient la paire clé/valeur avec la clé spécifiée.

```
bool ContainsKey(
   TKey  key     // clé
   );
```

Paramètres

key

[in] Clé.

Valeur de Retour

Retourne true si la table de hashage triée contient la pair clé/valeur ayant la clé et la valeur spécifiées, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

ContainsValue

Détermine si la table de hashage triée contient la paire clé/valeur avec la valeur spécifiée.

```
bool ContainsValue(
   TValue  value     // valeur
   );
```

Paramètres

value

[in] Valeur.

Valeur de Retour

Retourne true si la table de hashage triée contient la pair clé/valeur ayant la valeur spécifiée, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie toutes les paires clé/valeur de la table de hashage triée vers les tableaux spécifiés, en commençant à l'indice spécifié.

Version copiant une table de hashage vers le tableau de paires clé/valeur.

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // le tableau de destination key/valu
   const int                    dst_start=0      // l'indice de départ for writing
   );
```

Version copiant une table de hashage vers des tableaux de paires clé/valeur séparés.

```
int CopyTo(
   TKey&      dst_keys[],                        // le tableau de destination keys
   TValue&    dst_values[],                      // le tableau de destination values
   const int  dst_start=0                        // l'indice de départ for writing
   );
```

Paramètres

**&dst_array[]*

[out] Un tableau vers lequel toutes les paires de la table de hashage seront écrites.

&dst_keys[]

[out] Un tableau vers lequel toutes les clés de la table de hashage seront écrites.

&dst_values[]

[out] Un tableau vers lequel toutes les valeurs de la table de hashage seront écrites.

dst_start=0

[in] Un indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre de paires clé/valeurs copiées.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments de la table de hashage triée.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime la première occurrence de la paire clé/valeur de la table de hashage triée.

Version supprimant une paire clé/valeur basée sur la paire clé/valeur générée.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // la paire clé/valeur
   );
```

Version supprimant une paire clé/valeur basée sur la clé.

```
bool Remove(
   TKey  key                            // clé
   );
```

Paramètres

**item*

[in] La pair clé/valeur.

key

[in] Clé.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

TryGetValue

Retourne l'élément ayant la clé spécifiée dans une table de hashage triée.

```
bool TryGetValue(
   TKey     key,       // clé
   TValue&  value      // une variable pour écrire la valeur
   );
```

Paramètres

key

[in] Clé.

&value

[out] La variable dans laquelle la valeur spécifiée de la paire clé/valeur sera écrite.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

TrySetValue

Change la valeur de la paire clé/valeur de la table de hashage triée ayant la clé spécifiée.

```
bool TrySetValue(
   TKey    key,       // clé
   TValue  value      // nouvelle valeur
   );
```

Paramètres

key

[in] Clé.

value

[in] La nouvelle valeur à assigner à la paire clé/valeur spécifiée.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

CSortedSet<T>

CSortedSet<T> est une classe générique qui implémente l'interface ISet<T>.

Description

La classe CSortedSet<T> est une implémentation du set dynamique trié de données de type T, avec l'unicité de chaque valeur. Cette classe fournit les méthodes de base pour travailler avec les sets et toutes les opérations liées, telles que : l'union et l'intersection des sets, la définition de sous-ensembles stricts et non stricts, etc.

Déclaration

```
template<typename T>
class CSortedSet : public ISet<T>
```

Fichier header

```
#include <Generic\SortedSet.mqh>
```

Hiérarchie d'Héritage

ICollection
  ISet
    CSortedSet

Méthodes de classe

Méthode	Description
Add	Ajoute un élément à un set trié
Count	Retourne le nombre d'éléments dans un set trié
Contains	Détermine si un set trié contient un élément ayant la valeur spécifiée
Comparer	Retourne un pointeur sur l'interface IComparer<T>, utilisé pour organiser un set trié.
TryGetMin	Retourne l'élément minimum d'un set trié
TryGetMax	Retourne l'élément maximum d'un set trié
CopyTo	Copie tous les éléments d'un set trié dans le tableau spécifié en commençant à l'indice spécifié
Clear	Supprime tous les éléments d'un set trié
Remove	Supprime l'occurrence de l'élément spécifié du set trié
ExceptWith	Effectue l'opération de différence entre la collection courante et la collection passée en argument (tableau)
IntersectWith	Effectue l'opération d'intersection entre la collection courante et la collection passée en argument (tableau)

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
SymmetricExceptWith	Effectue l'opération de différence symétrique entre la collection courante et la collection passée en argument (tableau)
UnionWith	Effectue l'union entre la collection courante et la collection passée en argument (tableau)
IsProperSubsetOf	Détermine si le set trié actuel est un vrai sous-ensemble de la collection ou du tableau spécifié
IsProperSupersetOf	Détermine si le set trié actuel est un vrai sur-ensemble de la collection ou du tableau spécifié
IsSubsetOf	Détermine si le set trié actuel est un sous-ensemble de la collection ou du tableau spécifié
IsSupersetOf	Détermine si le set trié actuel est un sur-ensemble de la collection ou du tableau spécifié
Overlaps	Détermine si le set trié actuel chevauche la collection ou le tableau spécifié
SetEquals	Détermine si le set trié actuel contient tous les éléments de la collection ou du tableau spécifié
GetViewBetween	Retourne un sous-ensemble spécifié par les valeurs minimum et maximum du set trié actuel
GetReverse	Retourne une copie du set trié actuel dans lequel tous les éléments sont arrangés en ordre inverse

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élémént à un set trié.

```
bool Add(
   T  value     // la valeur de l'élément
   );
```

Paramètres

value

[in] La valeur de l'élément à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Count

Retourne le nombre d'éléments dans un set trié.

```
int Count();
```

Valeur de Retour

Retourne le nombre d'éléments.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si le set trié contient un élément ayant la valeur spécifiée.

```
bool Contains(
   T  item     // la valeur à chercher
   );
```

Paramètres

item

[in] La valeur à chercher.

Valeur de Retour

Retourne true si un élément ayant la valeur spécifié a été trouvé dans le set, false sinon.

© 2000-2024, MetaQuotes Ltd.

Comparer

Retourne un pointeur sur l'interface IComparer<T>, utilisé pour organiser un set trié.

```
IComparer<T>* Comparer() const;
```

Valeur de Retour

Retourne un pointeur sur l'interface IComparer<T>.

© 2000-2024, MetaQuotes Ltd.

TryGetMin

Retourne l'élément minimum du set trié.

```
bool TryGetMin(
   T&  min     // une variable pour écrire la valeur
   );
```

Paramètres

&min

[out] La variable dans laquelle la valeur minimum sera écrite.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

TryGetMax

Retourne l'élément maximum du set trié.

```
bool TryGetMax(
   T&  max     // une variable pour écrire la valeur
   );
```

Paramètres

&max

[out] La variable dans laquelle la valeur maximum sera écrite.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie tous les éléments d'un set trié dans le tableau spécifié en commençant à l'indice spécifié.

```
int CopyTo(
   T&          dst_array[],      // le tableau de destination
   const int   dst_start=0       // l'indice de départ for writing
   );
```

Paramètres

&dst_array[]

[out] Un tableau dans lequel les éléments du set seront écrits.

dst_start=0

[in] Un indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre d'éléments copiés.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments d'un set trié.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime l'occurrence de l'élément spécifié du set trié.

```
bool Remove(
   T  item     // l'élément par valeur
   );
```

Paramètres

item

[in] La valeur de l'élément à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

ExceptWith

Effectue l'opération de différence entre la collection courante et la collection passée en argument (tableau). Elle enlève de la collection courante (tableau) tous les éléments qui se trouvent dans la collection spécifiée (tableau).

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void ExceptWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection à exclure du set trié actuel.

&collection[]

[in] Un tableau à exclure du set trié actuel.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

IntersectWith

Effectue l'opération d'intersection entre la collection courante et la collection passée en argument (tableau). Elle modifie la collection courante pour ne contenir que les éléments qui se trouvent dans la collection spécifiée (tableau).

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void IntersectWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection avec laquelle opérer l'intersection avec le set actuel.

&collection[]

[in] Un tableau avec lequel opérer l'intersection avec le set actuel.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

SymmetricExceptWith

Effectue l'opération de différence symétrique entre la collection courante et la collection passée en argument (tableau). Elle modifie la collection courante pour ne contenir que les éléments qui se trouvent dans l'objet source ou dans la collection spécifiée (tableau) mais pas dans les deux.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void SymmetricExceptWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection avec laquelle produire la différence symétrique.

&collection[]

[in] Un tableau avec lequel produire la différence symétrique.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

UnionWith

Effectue l'union entre la collection courante et la collection passée en argument (tableau). Elle ajoute les éléments manquants de la collection spécifiée (tableau) dans la collection courante (array).

Version utilisant une collection implémentant l'interface ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // collection
   );
```

Version utilisant un tableau.

```
void UnionWith(
   T&  array[]                     // tableau
   );
```

Paramètres

**collection*

[in] Une collection avec laquelle opérer l'union avec le set actuel.

&collection[]

[in] Un tableau avec lequel opérer l'union avec le set actuel.

Note

Le résultat est écrit dans la collection courante (tableau).

© 2000-2024, MetaQuotes Ltd.

IsProperSubsetOf

Détermine si le set trié actuel est un vrai sous-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsProperSubsetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set trié courant est un vrai sous-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

IsProperSupersetOf

Détermine si le set trié actuel est un vrai sur-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsProperSupersetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set trié courant est un vrai sur-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

IsSubsetOf

Détermine si le set trié actuel est un sous-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsSubsetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set trié courant est un sous-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

IsSupersetOf

Détermine si le set trié actuel est un sur-ensemble de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // une collection pour déterminier la relation
   );
```

Version utilisant un tableau.

```
bool IsSupersetOf(
   T&  array[]                     // un tableau pour déterminer la relation
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer la relation.

&collection[]

[in] Un tableau pour déterminer la relation.

Valeur de Retour

Retourne true si le set trié courant est un sur-ensemble, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

Overlaps

Détermine si le set trié actuel chevauche la collection ou le tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // une collection à comparer
   );
```

Version utilisant un tableau.

```
bool Overlaps(
   T&  array[]                     // un tableau à comparer
   );
```

Paramètres

**collection*

[in] Une collection pour déterminer le chevauchement.

&collection[]

[in] Un tableau pour déterminer le chevauchement.

Valeur de Retour

Retourne true si le set trié actuel et une collection ou un tableau se chevauchent, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

SetEquals

Détermine si le set trié actuel contient tous les éléments de la collection ou du tableau spécifié.

Version utilisant une collection implémentant l'interface ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // une collection à comparer
   );
```

Version utilisant un tableau.

```
bool SetEquals(
   T&  array[]                     // un tableau à comparer
   );
```

Paramètres

**collection*

[in] Une collection d'éléments à comparer.

&collection[]

[in] Un tableau d'éléments à comparer.

Valeur de Retour

Retourne true si le set trié actuel contient tous les éléments de la collection ou du tableau spécifié, ou false sinon.

© 2000-2024, MetaQuotes Ltd.

GetViewBetween

Retourne un sous-ensemble spécifié par les valeurs minimum et maximum du set trié actuel.

```
bool GetViewBetween(
   T&  array[],          // le tableau de destination
   T   lower_value,      // la valeur minimale
   T   upper_value       // la valeur maximale
   );
```

Paramètres

&array[]

[out] Un tableau où écrire le sous-ensemble.

lower_value

[in] La valeur minimum de l'intervalle.

upper_value

[in] La valeur maximum de l'intervalle.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

GetReverse

Retourne une copie du set trié actuel dans lequel tous les éléments sont arrangés en ordre inverse.

```
bool GetReverse(
   T&  array[]     // le tableau de destination
   );
```

Paramètres

&array[]

[out] Un tableau où écrire.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

CStack<T>

CStack<T> est une classe générique qui implémente l'interface ICollection<T>.

Description

La classe CStack<T> est une collection dynamique de données de type T, organisée comme une pile opérant selon le principe LIFO (last in, first out).

Déclaration

```
template<typename T>
class CStack : public ICollection<T>
```

Fichier header

```
#include <Generic\Stack.mqh>
```

Hiérarchie d'Héritage

ICollection
  CStack

Méthodes de classe

Méthode	Description
Add	Ajoute un élément sur une pile
Count	Retourne le nombre d'éléments dans la pile
Contains	Détermine si une pile contient un élément ayant la valeur spécifiée
TrimExcess	Définit la capacité d'une pile au nombre actuel d'éléments
CopyTo	Copie tous les éléments d'une pile dans le tableau spécifié en commençant à l'indice spécifié
Clear	Supprime tous les éléments d'une pile
Remove	Supprime la première occurrence de l'élément spécifié dans la pile
Push	Ajoute un élément sur une pile
Peek	Retourne le premier élément d'une pile sans l'enlever
Pop	Retourne le premier élément de la pile et l'enlève

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un élément sur la pile.

```
bool Add(
   T  value     // la valeur de l'élément
   );
```

Paramètres

value

[in] La valeur de l'élément à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Count

Retourne le nombre d'éléments dans la pile.

```
int Count();
```

Valeur de Retour

Retourne le nombre d'éléments.

© 2000-2024, MetaQuotes Ltd.

Contains

Détermine si une pile contient un élément ayant la valeur spécifiée.

```
bool Contains(
   T  item     // la valeur à chercher
   );
```

Paramètres

item

[in] La valeur à chercher.

Valeur de Retour

Retourne true si un élément ayant la valeur spécifié a été trouvé dans la pile, false sinon.

© 2000-2024, MetaQuotes Ltd.

TrimExcess

Définit la capacité d'une pile au nombre actuel d'éléments, libérant ainsi la mémoire non utilisée.

```
void TrimExcess();
```

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copie tous les éléments d'une pile dans le tableau spécifié en commençant à l'indice spécifié.

```
int CopyTo(
   T&          dst_array[],      // le tableau de destination
   const int   dst_start=0       // l'indice de départ for writing
   );
```

Paramètres

&dst_array[]

[out] Un tableau dans lequel les éléments de la pile seront écrits.

dst_start=0

[in] Un indice du tableau à partir duquel la copie commence.

Valeur de Retour

Retourne le nombre d'éléments copiés.

© 2000-2024, MetaQuotes Ltd.

Clear

Supprime tous les éléments d'une pile.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime la première occurrence de l'élément spécifié dans la pile.

```
bool Remove(
   T  item     // l'élément par valeur
   );
```

Paramètres

item

[in] La valeur de l'élément à supprimer.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Push

Ajoute un élément sur la pile.

```
bool Push(
   T  value     // élément à ajouter
   );
```

Paramètres

value

[in] Un élément à ajouter.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Peek

Retourne le premier élément d'une pile sans l'enlever.

```
T Peek();
```

Valeur de Retour

Retourne le premier élément.

© 2000-2024, MetaQuotes Ltd.

Pop

Retourne le premier élément de la pile et l'enlève.

```
T Pop();
```

Valeur de Retour

Retourne le premier élément.

© 2000-2024, MetaQuotes Ltd.

ArrayBinarySearch

Recherche la valeur spécifiée dans un tableau unidimensionnel trié de façon ascendante en utilisant l'interface IComparable<T> pour comparer les éléments.

```
template<typename T>
int ArrayBinarySearch(
   T&              array[],          // tableau dans lequel effectuer la recherche
   const int       start_index,      // l'indice de départ
   const int       count,            // l'intervalle de recherche
   T               value,            // la valeur à chercher
   IComparer<T>*   comparer          // interface vers le comparateur
   );
```

Paramètres

&array[]

[out] Le tableau dans lequel effectuer la recherche.

value

[in] La valeur à chercher.

**comparer*

[in] Une interface pour comparer les éléments.

start_index

[in] L'indice de départ à partir duquel la recherche commence.

count

[in] La longueur de l'intervalle de recherche.

Valeur de Retour

Retourne l'indice de l'élément trouvé. Si la valeur recherchée n'est pas trouvée, retourne l'indice de l'élément le plus petit qui est le plus proche de la valeur.

© 2000-2024, MetaQuotes Ltd.

ArrayIndexOf

Recherche la première occurrence d'une valeur dans un tableau à 1 dimension.

```
template<typename T>
int ArrayIndexOf(
   T&          array[],          // tableau dans lequel effectuer la recherche
   T           value,            // la valeur à chercher
   const int   start_index,      // l'indice de départ
   const int   count             // l'intervalle de recherche
   );
```

Paramètres

&array[]

[out] Le tableau dans lequel effectuer la recherche.

value

[in] La valeur à chercher.

start_index

[in] L'indice de départ à partir duquel la recherche commence.

count

[in] La longueur de l'intervalle de recherche.

Valeur de Retour

Retourne l'indice du premier élément trouvé. Si la valeur n'est pas trouvée, retourne -1.

© 2000-2024, MetaQuotes Ltd.

ArrayLastIndexOf

Recherche la dernière occurrence d'une valeur dans un tableau à 1 dimension.

```
template<typename T>
int ArrayLastIndexOf(
   T&          array[],          // tableau dans lequel effectuer la recherche
   T           value,            // la valeur à chercher
   const int   start_index,      // l'indice de départ
   const int   count             // l'intervalle de recherche
   );
```

Paramètres

&array[]

[out] Le tableau dans lequel effectuer la recherche.

value

[in] La valeur à chercher.

start_index

[in] L'indice de départ à partir duquel la recherche commence.

count

[in] La longueur de l'intervalle de recherche.

Valeur de Retour

Retourne l'indice du dernier élément trouvé. Si la valeur n'est pas trouvée, retourne -1.

© 2000-2024, MetaQuotes Ltd.

ArrayReverse

Change l'ordre des éléments dans un tableau à 1 dimension.

```
template<typename T>
bool ArrayReverse(
   T&          array[],          // le tableau source
   const int   start_index,      // l'indice de départ
   const int   count             // le nombre d'éléments
   );
```

Paramètres

&array[]

[out] Le tableau source.

start_index

[in] L'indice de départ.

count

[in] Le nombre d'éléments du tableau participant à l'opération.

Valeur de Retour

Retourne true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

Compare

Compare les 2 valeurs, que l'une soit "supérieure, inférieure ou égale" à l'autre.

Version comparant deux valeurs de type bool.

```
int Compare(
   const bool  x,          // la première valeur
   const bool  y           // la deuxième valeur
   );
```

Version comparant deux valeurs de type char.

```
int Compare(
   const char  x,          // la première valeur
   const char  y           // la deuxième valeur
   );
```

Version comparant deux valeurs de type uchar.

```
int Compare(
   const uchar  x,         // la première valeur
   const uchar  y          // la deuxième valeur
   );
```

Version comparant deux valeurs de type short.

```
int Compare(
   const short  x,         // la première valeur
   const short  y          // la deuxième valeur
   );
```

Version comparant deux valeurs de type ushort.

```
int Compare(
   const ushort  x,        // la première valeur
   const ushort  y         // la deuxième valeur
   );
```

Version comparant deux valeurs de type color.

```
int Compare(
   const color  x,         // la première valeur
   const color  y          // la deuxième valeur
   );
```

Version comparant deux valeurs de type int.

```
int Compare(
   const int  x,           // la première valeur
   const int  y            // la deuxième valeur
   );
```

© 2000-2024, MetaQuotes Ltd.

Version comparant deux valeurs de type uint.

```
int Compare(
   const uint  x,          // la première valeur
   const uint  y           // la deuxième valeur
   );
```

Version comparant deux valeurs de type datetime.

```
int Compare(
   const datetime  x,      // la première valeur
   const datetime  y       // la deuxième valeur
   );
```

Version comparant deux valeurs de type long.

```
int Compare(
   const long  x,          // la première valeur
   const long  y           // la deuxième valeur
   );
```

Version comparant deux valeurs de type ulong.

```
int Compare(
   const ulong  x,         // la première valeur
   const ulong  y          // la deuxième valeur
   );
```

Version comparant deux valeurs de type float.

```
int Compare(
   const float  x,         // la première valeur
   const float  y          // la deuxième valeur
   );
```

Version comparant deux valeurs de type double.

```
int Compare(
   const double  x,        // la première valeur
   const double  y         // la deuxième valeur
   );
```

Version comparant deux valeurs de type string.

```
int Compare(
   const string  x,        // la première valeur
   const string  y         // la deuxième valeur
   );
```

Version comparant deux valeurs d'autres types.

```
template<typename T>
int Compare(
```

© 2000-2024, MetaQuotes Ltd.

```
    T   x,                  // la première valeur
    T   y                   // la deuxième valeur
    );
```

Paramètres

x

[in] La première valeur

y

[in] La deuxième valeur

Valeur de Retour

Retourne un nombre qui exprime le ratio de 2 valeurs comparées :

- si le résultat est inférieur à 0, x est inférieur à y (x<y)
- si le résultat est égal à 0, x est égal à y (x=y)
- si le résultat est supérieur à 0, x est plus grand que y (x>y)

Note

Si le type T est un objet qui implémente l'interface IComparable<T>, alors les objets seront comparés en utilisant la méthode Compare. Dans tous les autres cas, 0 est retourné.

© 2000-2024, MetaQuotes Ltd.

Equals

Compare l'égalité de 2 valeurs.

```
template<typename T>
bool Equals(
   T  x,      // la première valeur
   T  y       // la deuxième valeur
   );
```

Paramètres

x

[in] La première valeur

y

[in] La deuxième valeur

Valeur de Retour

Retourne true si les objets sont égaux, false sinon.

Note

Si le type T est un objet qui implémente l'interface IEqualityComparable<T>, alors les objets seront comparés en utilisant sa méthode de comparaison Equals. La comparaison standard d'égalité est utilisée dans tous les autres cas.

© 2000-2024, MetaQuotes Ltd.

GetHashCode

Calcule la valeur du code de hashage.

Version utilisant le type bool.

```
int GetHashCode(
   const bool  value          // valeur
   );
```

Version utilisant le type char.

```
int GetHashCode(
   const char  value          // valeur
   );
```

Version utilisant le type uchar.

```
int GetHashCode(
   const uchar  value         // valeur
   );
```

Version utilisant le type short.

```
int GetHashCode(
   const short  value         // valeur
   );
```

Version utilisant le type ushort.

```
int GetHashCode(
   const ushort  value        // valeur
   );
```

Version utilisant le type color.

```
int GetHashCode(
   const color  value         // valeur
   );
```

Version utilisant le type int.

```
int GetHashCode(
   const int  value           // valeur
   );
```

Version utilisant le type uint.

```
int GetHashCode(
   const uint  value          // valeur
   );
```

Version utilisant le type datetime.

```
int GetHashCode(
```

© 2000-2024, MetaQuotes Ltd.

```
   const datetime  value     // valeur
   );
```

Version utilisant le type long.

```
int GetHashCode(
   const long  value         // valeur
   );
```

Version utilisant le type ulong.

```
int GetHashCode(
   const ulong  value        // valeur
   );
```

Version utilisant le type float.

```
int GetHashCode(
   const float  value        // valeur
   );
```

Version utilisant le type double.

```
int GetHashCode(
   const double  value       // valeur
   );
```

Version utilisant le type string.

```
int GetHashCode(
   const string  value       // valeur
   );
```

Version utilisant d'autres types.

```
template<typename T>
int GetHashCode(
   T  value                  // valeur
   );
```

Paramètres

value

[in] La valeur pour laquelle vous souhaitez calculer le code de hashage.

Valeur de Retour

Retourne le code de hashage.

Note

Si le type T est un objet qui implémente l'interface IEqualityComparable<T>, alors le code de hashage sera obtenu en utilisant sa méthode HashCode. Dans tous les autres cas, le code de hashage sera calculé comme la valeur de hashage du nom du type de la valeur.

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Opérations sur les Fichiers

Cette section contient les détails techniques d'utilisation des classes d'opérations sur les fichiers et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Utiliser les classes liées aux opérations sur les fichiers vous fera gagner du temps lors du développement d'applications qui utilisent les opérations d'entrée/sortie sur les fichiers.

La Bibliothèque Standard MQL5 (en terme de données) est située dans le répertoire de travail du terminal dans le répertoire Include\Files.

Classe	Description
CFile	Classe de base pour les opérations sur les fichiers
CFileBin	Classe pour les opérations sur les fichiers binaires
CFileTxt	Classe pour les opérations sur les fichiers textes

© 2000-2024, MetaQuotes Ltd.

CFile

CFile est la classe de base pour les classes CFileBin et CFileTxt.

Description

La classe CFile fournit un accès simplifié pour toutes ses classes dérivées aux fonctions relatives aux fichiers et aux répertoires de l'API MQL5.

Déclaration

```
class CFile: public CObject
```

Titre

```
#include <Files\File.mqh>
```

Hiérarchie d'héritage

CObject
  CFile

Descendants directs

CFileBin, CFilePipe, CFileTxt

Méthodes de Classe

Attributs	
Handle	Retourne le handle du fichier
Filename	Retourne le nom du fichier
Flags	Retourne les flags du fichier
SetUnicode	Définit/Réinitialise le flag FILE_UNICODE
SetCommon	Définit/Réinitialise le flag FILE_COMMON
Méthodes générales liées aux fichiers	
Open	Ouvre un fichier
Close	Ferme un fichier
Delete	Supprime un fichier
IsExist	Vérifie l'existence d'un fichier
Copy	Copie un fichier
Move	Renomme/déplace un fichier
Size	Retourne la taille du fichier

© 2000-2024, MetaQuotes Ltd.

Attributs	
Tell	Retourne la position courante dans le fichier
Seek	Définit la position courante dans le fichier
Flush	Ecrit les données sur le disque
IsEnding	Vérifie la fin du fichier
IsLineEnding	Vérifie la fin de ligne du fichier
Méthodes générales liées aux répertoires	
FolderCreate	Crée un répertoire
FolderDelete	Supprime un répertoire
FolderClean	Vide un répertoire
Méthodes de recherche	
FileFindFirst	Commence une recherche de fichier
FileFindNext	Continue une recherche de fichier
FileFindClose	Ferme la handle de recherche

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Handle

Retourne le handle du fichier ouvert.

```
int  Handle()
```

Valeur de retour

Handle du fichier ouvert assigné à l'instance de classe. Si aucun fichier n'est assigné, retourne -1.

© 2000-2024, MetaQuotes Ltd.

FileName

Retourne le nom du fichier ouvert.

```
string  FileName()
```

Valeur de retour

Nom du fichier ouvert assigné à l'instance de la classe. Si aucun fichier n'est assigné, retourne "".

© 2000-2024, MetaQuotes Ltd.

Flags

Retourne les flags du fichier ouvert.

```
int  Flags()
```

Valeur de retour

Flags du fichier ouvert assigné à l'instance de la clase.

© 2000-2024, MetaQuotes Ltd.

SetUnicode

Définit/Réinitialise le flag FILE_UNICODE.

```
void  SetUnicode(
   bool  unicode      // Nouvelle valeur du flag
   )
```

Paramètres

unicode

[in] Nouvelle valeur du flag FILE_UNICODE.

Note

Le résultat des opérations sur les chaînes de caractères dépend du flag FILE_UNICODE. S'il est faux, les codes ANSI sont utilisés (symboles sur 1 octet). S'il est défini, les codes UNICODE sont utilisés (symboles sur 2 octets). Si le fichier a déjà été ouvert, le flag ne peut pas être changé.

© 2000-2024, MetaQuotes Ltd.

SetCommon

Définit/Réinitialise le flag FILE_COMMON.

```
void  SetCommon(
   bool  common      // Nouvelle valeur du flag
   )
```

Paramètres

common

[in] Nouvelle valeur du flag FILE_COMMON.

Note

La flag FILE_UNICODE détermine le répertoire courant de travail. S'il est faux, le répertoire local du terminal est utilisé comme répertoire courant de travail. S'il est vrai, le répértoire général est utilisé comme répertoire courant de travail. Si le fichier a déjà été ouvert, le flag ne peut pas être changé.

© 2000-2024, MetaQuotes Ltd.

Open

Ouvre le fichier spécifié et l'assigne à l'instance de class en cas de succès.

```
int  Open(
   const string  file_name,        // nom du fichier
   int           flags,            // Flags
   short         delimiter=9       // Séparateur
   )
```

Paramètres

file_name

[in] Nom du fichier à ouvrir.

flags

[in] Flags d'ouverture du fichier.

delimiter=9

[in] Séparateur du fichier CSV.

Valeur de retour

Handle sur le fichier ouvert.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2024, MetaQuotes Ltd.

Close

Ferme le fichier assigné à l'instance de la classe.

```
void  Close()
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime le fichier assigné à l'instance du fichier.

```
void  Delete()
```

Delete

Supprime le fichier spécifié.

```
void  Delete(
   const string  file_name      // Nom du fichier
   )
```

Paramètres

file_name

[in] Nom du fichier à supprimer.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2024, MetaQuotes Ltd.

IsExist

Vérifie l'existence d'un fichier

```
bool  IsExist(
   const string  file_name      // Nom du fichier
   )
```

Paramètres

file_name

[in] Nom du fichier à vérifier.

Valeur de retour

vrai si le fichier existe.

© 2000-2024, MetaQuotes Ltd.

Copy

Copie un fichier.

```
bool  Copy(
   const string  src_name,       // Nom du fichier source
   int           src_flag,       // Flag
   const string  dst_name,       // Nom du fichier destination
   int           dst_flags       // Flags
   )
```

Paramètres

src_name

[in] Nom du fichier source à copier.

src_flag

[in] Flags du fichier à copier (seul FILE_COMMON est utilisé).

dst_name

[in] Nom du fichier de destination.

dst_flags

[in] Flags du fichier de destination (seuls FILE_REWRITE et FILE_COMMON sont utilisés).

Valeur de retour

vrai si réalisé avec succès, faux si la copie a échoué.

© 2000-2024, MetaQuotes Ltd.

Move

Renomme/déplace un fichier.

```
bool  Move(
   const string  src_name,       // Nom du fichier source
   int           src_flag,       // Flag
   const string  dst_name,       // Nom du fichier destination
   int           dst_flags       // Flags
   )
```

Paramètres

src_name

[in] Nom du fichier à déplacer.

src_flag

[in] Flags du fichier à copier (seul FILE_COMMON est utilisé).

dst_name

[in] Nom du fichier de destination.

dst_flags

[in] Flags du fichier de destination (seuls FILE_REWRITE et FILE_COMMON sont utilisés).

Valeur de retour

vrai si réalisé avec succès, faux si le déplacement du fichier a échoué.

© 2000-2024, MetaQuotes Ltd.

Size

Retourne la taille du fichier en octets.

```
ulong  Size()
```

Valeur de retour

Taille du fichier en octets. Si aucun fichier n'est assigné, retourne ULONG_MAX.

© 2000-2024, MetaQuotes Ltd.

Tell

Retourne la position courante dans le fichier.

```
ulong  Tell()
```

Valeur de retour

La position courante dans le fichier. Si aucun fichier n'est assigné, retourne ULONG_MAX.

© 2000-2024, MetaQuotes Ltd.

Seek

Définit la position courante dans le fichier.

```
void  Seek(
   long                offset,      // Décalage
   ENUM_FILE_POSITION  origin       // Origine
   )
```

Paramètres

offset

[in] Décalage dans le fichier en octets (peut être négatif).

origin

[in] Origine du décalage.

Valeur de retour

vrai si réalisé avec succès, faux si la position dans le fichier n'a pas été changée.

© 2000-2024, MetaQuotes Ltd.

Flush

Ecrit (vide le tampon) toutes les données du buffer du fichier en entrée/sortie sur le disque.

```
void  Flush()
```

© 2000-2024, MetaQuotes Ltd.

IsEnding

Vérifie la fin du fichier. Utilisé pendant les opérations de lecture du fichier.

```
bool  IsEnding()
```

Valeur de retour

vrai si la fin du fichier est atteinte après la dernière lecture ou opération de recherche.

© 2000-2024, MetaQuotes Ltd.

IsLineEnding

Vérifie la fin de ligne du fichier. Utilisé pendant les opérations de lecture du fichier.

```
bool  IsLineEnding()
```

Valeur de retour

vrai si la fin de ligne du fichier est atteinte après la dernière opération de lecture d'un fichier txt ou csv (caractères CR-LF).

© 2000-2024, MetaQuotes Ltd.

FolderCreate

Crée un nouveau répertoire.

```
bool  FolderCreate(
   const string  folder_name      // Nom du répertoire
   )
```

Paramètres

folder_name

[in] Nom du répertoire à créer. Contient le chemin relatif du répertoire définit avec le flag FILE_COMMON.

Valeur de retour

vrai si réalisé avec succès, faux si le répertoire n'a pas été créé.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2024, MetaQuotes Ltd.

FolderDelete

Supprime le répertoire spécifié.

```
bool  FolderDelete(
   const string  folder_name      // Nom du répertoire
   )
```

Paramètres

folder_name

[in] Nom du répertoire à supprimer. Contient le chemin relatif du répertoire définit avec le flag FILE_COMMON.

Valeur de retour

vrai si réalisé avec succès, faux si le répertoire n'a pas été supprimé.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2024, MetaQuotes Ltd.

FolderClean

Vide le répertoire sépcifié.

```
bool  FolderClean(
   const string  folder_name      // Nom du répertoire
   )
```

Paramètres

folder_name

[in] Nom du répertoire à supprimer. Contient le chemin relatif du répertoire définit avec le flag FILE_COMMON.

Valeur de retour

vrai si réalisé avec succès, faux si le répertoire n'a pas été vidé.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2024, MetaQuotes Ltd.

FileFindFirst

Commence la recherche de fichier en utilisant le filtre spécifié.

```
int  FileFindFirst(
   const string  filter,        // Filtre de recherche
   string&       file_name      // Référence sur une chaîne de caractères
   )
```

Paramètres

filter

[in] Filtre de recherche.

file_name

[out] Référence sur une chaîne de caractères pour le premier fichier trouvé.

Valeur de retour

En cas de succès, retourne le handle qui peut être utilisé pour continuer la recherche de fichier en utilisant FileFindNext, ou INVALID_HANDLE si aucun fichier ne correspond au filtre spécifié.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2024, MetaQuotes Ltd.

FileFindNext

Continue la recherche, commencée avec la fonction FileFindFirst().

```
bool  FileFindNext(
   int      search_handle,     // Handle de recherche
   string&  file_name          // Référence sur la chaîne de caractères pour le procha
   )
```

Paramètres

search_handle

[in] Handle de recherche, retourné par la méthode FileFindFirst().

file_name

[out] Référence sur la chaîne de caractères si un fichier est trouvé.

Valeur de retour

vrai en cas de succès, faux si aucun fichier correspondant au filtre spécifié n'est trouvé.

© 2000-2024, MetaQuotes Ltd.

FileFindClose

Ferme la handle du fichier.

```
void  FileFindClose(
   int  search_handle      // Handle de recherche
   )
```

Paramètres

search_handle

[in] Handle de recherche, retourné par la méthode FileFindFirst().

© 2000-2024, MetaQuotes Ltd.

CFileBin

CFileBin est une classe permettant un accès simplifié aux fichiers binaires.

Description

La classe CFileBin fournit un accès aux fichiers binaires.

Déclaration

```
class CFileBin: public CFile
```

Titre

```
#include <Files\FileBin.mqh>
```

Hiérarchie d'héritage

CObject
  CFile
    CFileBin

Méthodes de Classe

Méthodes d'ouverture	
Open	Ouvre un fichier binaire
Méthodes d'écriture	
WriteChar	Ecrit une variable de type char ou uchar
WriteShort	Ecrit une variable de type short ou ushort
WriteInteger	Ecrit une variable de type int ou uint
WriteLong	Ecrit une variable de type long ou ulong
WriteFloat	Ecrit une variable de type float
WriteDouble	Ecrit une variable de type double
WriteString	Ecrit une variable de type string
WriteCharArray	Ecrit un tableau de variables de type char ou uchar
WriteShortArray	Ecrit un tableau de variables de type short ou ushort
WriteIntegerArray	Ecrit un tableau de variables de type int ou uint
WriteLongArray	Ecrit un tableau de variables de type long ou ulong
WriteFloatArray	Ecrit un tableau de variables de type float
WriteDoubleArray	Ecrit un tableau de variables de type double
WriteObject	Ecrit les données d'une instance dérivant de la classe CObject

© 2000-2024, MetaQuotes Ltd.

Méthodes d'ouverture	
Méthodes de lecture	
ReadChar	Lit une variable de type char ou uchar
ReadShort	Lit une variable de type short ou ushort
ReadInteger	Lit une variable de type int ou uint
ReadLong	Lit une variable de type long ou ulong
ReadFloat	Lit une variable de type float
ReadDouble	Lit une variable de type double
ReadString	Lit une variable de type string
ReadCharArray	Lit un tableau de variables de type char ou uchar
ReadShortArray	Lit un tableau de variables de type short ou ushort
ReadIntegerArray	Lit un tableau de variables de type int ou uint
ReadLongArray	Lit un tableau de variables de type long ou ulong
ReadFloatArray	Lit un tableau de variables de type float
ReadDoubleArray	Lit un tableau de variables de type double
ReadObject	Lit les données d'une instance de classe dérivant de la classe CObject

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CFile

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

© 2000-2024, MetaQuotes Ltd.

Open

Ouvre le fichier binaire spécifié et l'assigne à l'instance de classe en cas de succès.

```
int  Open(
   const string  file_name,      // Nom du fichier
   int           flags           // Flags
   )
```

Paramètres

file_name

[in] Nom du fichier à ouvrir.

flags

[in] Flags d'ouverture du fichier (flag FILE_BIN).

Valeur de retour

Handle sur le fichier ouvert.

© 2000-2024, MetaQuotes Ltd.

WriteChar

Ecrit une variable de type char ou uchar dans le fichier.

```
uint  WriteChar(
   char  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteShort

Ecrit une variable de type short ou ushort dans le fichier.

```
uint  WriteShort(
   short  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteInteger

Ecrit une variable de type int ou uint dans le fichier.

```
uint  WriteInteger(
   int  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteLong

Ecrit une variable de type long ou ulong dans le fichier.

```
uint  WriteLong(
   long  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteFloat

Ecrit une variable de type float dans le fichier.

```
uint  WriteFloat(
   float  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteDouble

Ecrit une variable de type double dans le fichier.

```
uint  WriteDouble(
   double  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteString

Ecrit une variable de type string dans le fichier.

```
uint  WriteString(
   const string  value      // Valeur
   )
```

Paramètres

value

[in] Chaîne de caractères à écrire.

Valeur de retour

Nombre d'octets écrits.

WriteString

Ecrit une variable de type string dans le fichier.

```
uint  WriteString(
   const string  value,     // Valeur
   int           size       // Taille
   )
```

Paramètres

value

[in] Chaîne de caractères à écrire.

size

[in] Nombre d'octets à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteCharArray

Ecrit un tableau de variables de type char ou uchar dans le fichier.

```
uint  WriteCharArray(
   char&  array[],            // Référence sur le tableau
   int    start_item=0,       // Elément de départ
   int    items_count=-1      // Number of elements
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 - pour le tableau entier).

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteShortArray

Ecrit un tableau de variables de type short ou ushort dans le fichier.

```
uint  WriteShortArray(
   short&  array[],            // Tableau à écrire
   int     start_item=0,       // Elément de départ
   int     items_count=-1      // Nombre d'éléments à écrite (-1 pour le tableau compl
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 - pour le tableau entier).

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteIntegerArray

Ecrit un tableau de variables de type int ou uint dans le fichier.

```
uint  WriteIntegerArray(
   int&  array[],            // Tableau à écrire
   int   start_item=0,       // Elément de départ
   int   items_count=-1      // Nombre d'éléments à écrite (-1 pour le tableau complet
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 - pour le tableau entier).

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteLongArray

Ecrit un tableau de variables de type long ou <t3>ulong</t3><t4> dans le fichier.</t4>

```
uint  WriteLongArray(
   long&  array[],            // Tableau à écrire
   int    start_item=0,       // Elément de départ
   int    items_count=-1      // Nombre d'éléments à écrite (-1 pour le tableau comple
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 pour le tableau complet). (-1 - for whole array).

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteFloatArray

Ecrit un tableau de variables de type float dans le fichier.

```
uint  WriteFloatArray(
   float&  array[],            // Tableau à écrire
   int     start_item=0,       // Elément de départ
   int     items_count=-1      // Nombre d'éléments à écrite (-1 pour le tableau compl
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 pour le tableau complet). (-1 - for whole array).

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteDoubleArray

Ecrit un tableau de variables de type double dans le fichier.

```
uint  WriteDoubleArray(
   double&  array[],            // Tableau à écrire
   int      start_item=0,       // Elément de départ
   int      items_count=-1      // Nombre d'éléments à écrite (-1 pour le tableau comp
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 pour le tableau complet). (-1 - for whole array).

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

WriteObject

Ecrit les données d'une instance de classe dérivant de la classe CObject dans le fichier.

```
bool  WriteObject(
   CObject*  object      // Référence sur l'objet
   )
```

Paramètres

object

[in] Référence sur l'instance à écrire de la classe dérivant de la classe CObject.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été écrite.

© 2000-2024, MetaQuotes Ltd.

ReadChar

Lit une variable de type char ou uchar depuis le fichier.

```
bool  ReadChar(
   char&  value      // Variable destination
   )
```

Paramètres

value

[in] Variable destination de type char.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadShort

Lit une variable de type short ou ushort depuis le fichier.

```
bool  ReadShort(
   short&  value
   )
```

Paramètres

value

[in] Variable destination de type short ou ushort.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadInteger

Lit une variable de type int ou uint depuis le fichier.

```
bool  ReadInteger(
   int&  value      // Variable destination
   )
```

Paramètres

value

[in] Variable destination de type int ou uint.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadLong

Lit une variable de type long ou ulong depuis le fichier.

```
bool  ReadLong(
   long&  value
   )
```

Paramètres

value

[in] Variable destination de type long ou ulong.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadFloat

Lit une variable de type float depuis le fichier.

```
bool  ReadFloat(
   float&  value      // Variable destination
   )
```

Paramètres

value

[in] Variable destination de type float.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadDouble

Lit une variable de type double depuis le fichier.

```
bool  ReadDouble(
   double&  value
   )
```

Paramètres

value

[in] Variable destination de type double.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadString

Lit une variable de type string depuis le fichier.

```
bool  ReadString(
   string&  value      // Chaîne desintation
   )
```

Paramètres

value

[in] Variable destination de type string.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

ReadString

Lit une variable de type string depuis le fichier.

```
bool  ReadString(
   string&  value
   )
```

Paramètres

value

[in] Variable destination de type string.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadCharArray

Lit un tableau de variables de type char ou uchar depuis le fichier.

```
bool  ReadCharArray(
   char&  array[],            // Tableau destination
   int    start_item=0,       // Elément de départ
   int    items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type char ou uchar.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadShortArray

Lit un tableau de variables de type short ou ushort depuis le fichier.

```
bool  ReadShortArray(
   short&  array[],            // Tableau destination
   int     start_item=0,       // Elément de départ
   int     items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type short ou ushort.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadIntegerArray

Lit un tableau de variables de type int ou uint depuis le fichier.

```
bool  ReadIntegerArray(
   int&  array[],            // Tableau destination
   int   start_item=0,       // Elément de départ
   int   items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type int ou uint.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadLongArray

Lit un tableau de variables de type long ou ulong depuis le fichier.

```
bool  ReadLongArray(
   long&  array[],            // Tableau destination
   int    start_item=0,       // Elément de départ
   int    items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type long ou ulong.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadFloatArray

Lit un tableau de variables de type float depuis le fichier.

```
bool  ReadFloatArray(
   float&  array[],            // Tableau destination
   int     start_item=0,       // Elément de départ
   int     items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type float.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadDoubleArray

Lit un tableau de variables de type double depuis le fichier.

```
bool  ReadDoubleArray(
   double&  array[],            // Tableau destination
   int      start_item=0,       // Elément de départ
   int      items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type double.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

ReadObject

Lit les données d'une instance de classe dérivant de la classe CObject depuis le fichier.

```
bool  ReadObject(
   CObject*  object      // Référence sur l'objet
   )
```

Paramètres

object

[in] Référence sur l'instance à lire de la classe dérivant de la classe CObject.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2024, MetaQuotes Ltd.

CFileTxt

CFileTxt est une classe permettant un accès simplifié aux fichiers textes.

Description

La classe CFileTxt fournit un accès aux fichiers textes.

Déclaration

```
class CFileTxt: public CFile
```

Titre

```
#include <Files\FileTxt.mqh>
```

Hiérarchie d'héritage

COBJECT
  CFile
    CFileTxt

Méthodes de Classe

Méthodes d'ouverture	
Open	Ouvre un fichier texte
Méthodes d'écriture	
WriteString	Ecrit une variable de type string dans le fichier
Méthodes de lecture	
ReadString	Lit une variable de type string depuis le fichier.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CFile

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

© 2000-2024, MetaQuotes Ltd.

Open

Ouvre le fichier texte spécifié et l'assigne à l'instance de classe en cas de succès.

```
int  Open(
   const string  file_name,      // nom du fichier
   int           flags           // flags
   )
```

Paramètres

file_name

[in] Nom du fichier à ouvrir.

flags

[in] Flags d'ouverture du fichier (flag FILE_TXT).

Valeur de retour

Handle du fichier ouvert.

© 2000-2024, MetaQuotes Ltd.

WriteString

Ecrit une variable de type string dans le fichier.

```
uint  WriteString(
   const string  value      // Chaîne de caractères à écrire
   )
```

Paramètres

value

[in] Chaîne de caractères à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2024, MetaQuotes Ltd.

ReadString

Lit une variable de type string depuis le fichier.

```
string  ReadString()
```

Valeur de retour

Chaîne de caractères qui a été lue.

© 2000-2024, MetaQuotes Ltd.

Opérations sur les chaînes de caractères

Cette section contient les détails techniques d'utilisation des classes d'opérations sur les chaînes de caractères et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Utiliser les classes liées aux opérations sur les chaînes de caractères vous fera gagner du temps lors du développement d'applications qui utilisent les opérations de traitement de chaînes de caractères.

La Bibliothèque Standard MQL5 est située dans le répertoire de travail du terminal dans le répertoire Include\Strings.

Classe	Description
CString	Classe pour les opérations sur les chaines de caractères

© 2000-2024, MetaQuotes Ltd.

CString

CString est une classe permettant un accès simplifié aux variables de type string.

Description

La classe CFile fournit un accès simplifié à tous ses descendants aux fonctions de chaînes de caractères de l'API MQL5.

Déclaration

```
class CString: public CObject
```

Titre

```
#include <Strings\String.mqh>
```

Hiérarchie d'héritage

CObject
  CString

Méthodes de Classe

Méthodes d'accès aux données	
Str	Retourne la chaîne de caractères
Len	Retourne la longueur de la chaîne de caractères
Copy	Copie une chaîne de caractères
Méthodes de remplissage	
Fill	Remplit une chaîne de caractères avec un caractère spécifié
Assign	Assigne une chaîne de caractères
Append	Ajoute une chaîne de caractères
Insert	Insère une chaîne de caractères
Méthodes de comparaison	
Compare	Compare une chaîne de caractères
CompareNoCase	Compare une chaîne de caractères sans tenir compte de la casse
Méthodes de sous-chaînes de caractères	
Left	Retourne le nombre spécifié de caractères depuis la gauche de la chaîne de caractères

© 2000-2024, MetaQuotes Ltd.

Méthodes d'accès aux données	
Right	Retourne le nombre spécifié de caractères depuis la droite de la chaîne de caractères
Mid	Retourne le nombre spécifié de caractères de la chaîne de caractères
Méthodes d'effacement/suppression	
Trim	Supprime tous les ensembles de caractères spécifiés devant et derrière la chaîne de caractères
TrimLeft	Supprime tous les ensembles de caractères spécifiés devant la chaîne de caractères
TrimRight	Supprime tous les ensembles de caractères spécifiés derrière la chaîne de caractères
Clear	Vide la chaîne de caractères
Méthodes de conversion	
ToUpper	Transforme une chaîne de caractères en majuscules.
ToLower	Transforme une chaîne de caractères en minuscules.
Reverse	Retourne une chaîne de caractères
Méthodes de recherche	
Find	Recherche la première occurence d'une chaîne de caractères
FindRev	Recherche la dernière occurence d'une chaîne de caractères
Remove	Supprime une sous-chaîne de caractères de la chaîne de caractères
Replace	Remplace une sous-chaîne de caractères

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type

© 2000-2024, MetaQuotes Ltd.

Str

Retourne la chaîne de caractères.

```
string  Str() const;
```

Valeur de retour

Copie de la chaîne de caractères.

© 2000-2024, MetaQuotes Ltd.

Len

Retourne la longueur de la chaîne de caractères.

```
uint  Len() const;
```

Valeur de retour

Longueur de la chaîne de caractères.

© 2000-2024, MetaQuotes Ltd.

Copy

Copie une chaîne de caractères par référence.

```
void  Copy(
   string&  copy      // Référence
   ) const;
```

Paramètres

copy

[in] Référence sur une chaîne de caractères à copier.

Copy

Copie une chaîne de caractères dans l'instance de classe CString.

```
void  Copy(
   CString*  copy      // Descripteur de l'objet
   ) const;
```

Paramètres

copy

[in] Descripteur de l'objet de classe CString.

© 2000-2024, MetaQuotes Ltd.

Fill

Remplit une chaîne de caractères avec le caractère spécifié.

```
bool  Fill(
   short  character      // Caractère
   )
```

Paramètres

character

[in] Caractère de remplissage.

Valeur de retour

vrai si réalisé avec succès, faux si la chaîne de caractères n'a pas pu être remplie.

© 2000-2024, MetaQuotes Ltd.

Assign

Assigne une chaîne de caractères.

```
void  Assign(
   const string  str      // Chaîne de caractères à assigner
   )
```

Paramètres

str

[in] Chaîne de caractères à assigner.

Assign

AjotueAssigne une chaîne de caractères à l'instance de classe CString.

```
void  Assign(
   CString*  str      // Descripteur de l'objet
   )
```

Paramètres

str

[in] Descripteur de l'objet de classe CString à assigner.

© 2000-2024, MetaQuotes Ltd.

Append

Ajoute une chaîne de caractères à la fin.

```
void  Append(
   const string  str      // Chaîne de caractères à ajouter
   )
```

Paramètres

str

[in] Chaîne de caractères à ajouter.

Append

Ajoute une chaîne de caractères à l'instance de classe CString.

```
void  Append(
   CString*  string      // Descripteur de l'objet
   )
```

Paramètres

string

[in] Descripteur de l'objet de classe CString à ajouter.

© 2000-2024, MetaQuotes Ltd.

Insert

Insère une chaîne de caractères à la position spécifiée.

```
uint  Insert(
   uint          pos,      // Position
   const string  str       // Chaîne de caractères à insérer
   )
```

Paramètres

pos

[in] Position d'insertion.

str

[in] Chaîne de caractères à insérer.

Valeur de retour

Taille de la chaîne de caractères résultante.

Insert

Insère une chaîne de caractères à la position spécifiée dans l'instance de classe CString.

```
uint  Insert(
   uint      pos,      // Position
   CString*  str       // Descripteur de l'objet
   )
```

Paramètres

pos

[in] Position d'insertion.

str

[in] Descripteur de l'objet de classe CString à insérer.

Valeur de retour

Taille de la chaîne de caractères résultante.

© 2000-2024, MetaQuotes Ltd.

Compare

Compare une chaîne de caractères.

```
int  Compare(
   const string  str      // Chaîne de caractères à comparer
   ) const;
```

Paramètres

str

[in] Chaîne de caractères à comparer.

Valeur de retour

Retourne 0 si les chaînes de caractères sont identiques, -1 si la chaîne de caractères de la classe est inférieure à la chaîne de comparaison, 1 si la chaîne de caractères de la classe est supérieure à la chaîne de comparaison.

Compare

Compare une chaîne de caractères avec une chaîne de caractères de l'instance de class CString.

```
int  Compare(
   CString*  str      // Descripteur de l'objet
   ) const;
```

Paramètres

str

[in] Descripteur de l'objet de classe CString à comparer.

Valeur de retour

Retourne 0 si les chaînes de caractères sont identiques, -1 si la chaîne de caractères de la classe est inférieure à la chaîne de comparaison, 1 si la chaîne de caractères de la classe est supérieure à la chaîne de comparaison.

© 2000-2024, MetaQuotes Ltd.

CompareNoCase

Compare une chaîne de caractères sans tenir compte de la casse.

```
int  CompareNoCase(
   const string  str      // Chaîne de caractères à comparer
   ) const;
```

Paramètres

str

[in] Chaîne de caractères à comparer.

Valeur de retour

Retourne 0 si les chaînes de caractères sont identiques, -1 si la chaîne de caractères de la classe est inférieure à la chaîne de comparaison, 1 si la chaîne de caractères de la classe est supérieure à la chaîne de comparaison.

CompareNoCase

Compare une chaîne de caractères (sans tenir compte de la casse) avec une chaîne de caractères de l'instance de class CString.

```
int  CompareNoCase(
   CString*  str      // Descripteur de l'objet
   ) const;
```

Paramètres

str

[in] Descripteur de l'objet de classe CString à comparer.

Valeur de retour

Retourne 0 si les chaînes de caractères sont identiques, -1 si la chaîne de caractères de la classe est inférieure à la chaîne de comparaison, 1 si la chaîne de caractères de la classe est supérieure à la chaîne de comparaison.

© 2000-2024, MetaQuotes Ltd.

Left

Retourne le nombre spécifié de caractères depuis la gauche de la chaîne de caractères.

```
string  Left(
   uint  count      // Nombre de caractères
   )
```

Paramètres

count

[in] Nombre de caractères.

Valeur de retour

Chaîne de caractères résultante.

© 2000-2024, MetaQuotes Ltd.

Right

Retourne le nombre spécifié de caractères depuis la droite de la chaîne de caractères.

```
string  Right(
   uint  count      // Nombre de caractères
   )
```

Paramètres

count

[in] Nombre de caractères.

Valeur de retour

Chaîne de caractères résultante.

© 2000-2024, MetaQuotes Ltd.

Mid

Retourne le nombre spécifié de caractères de la chaîne de caractères.

```
string  Mid(
   uint  pos,        // Position
   uint  count       // Nombre de caractères
   )
```

Paramètres

pos

[in] Position de la chaîne de caractères.

count

[in] Nombre de caractères.

Valeur de retour

Chaîne de caractères résultante.

© 2000-2024, MetaQuotes Ltd.

Trim

Supprime toutes les occurences d'un ensemble de caractères spécifiés (et ' ','\t','\r','\n') avant et après la chaîne de caractères.

```
int  Trim(
   const string  targets      // Ensemble de caractères à supprimer
   )
```

Paramètres

targets

[in] Ensemble de caractères à supprimer.

Valeur de retour

Nombre de caractères supprimés.

Exemple :

```
//--- exemple d'utilisation de CString::Trim
#include <Strings\String.mqh>
//---
void OnStart()
  {
   CString str;
   //---
   str.Assign("   \t\tABCD\r\n");
   printf("Source string '%s'",str.Str());
   //---
   str.Trim("DA-DA-DA");
   printf("Result string '%s'",str.Str());
  }
```

© 2000-2024, MetaQuotes Ltd.

TrimLeft

Supprime toutes les occurences d'un ensemble de caractères spécifiés (et ' ','\t','\r','\n') au début de la chaîne de caractères.

```
int  TrimLeft(
   const string  targets      // Ensemble de caractères à supprimer
   )
```

Paramètres

targets

[in] Ensemble de caractères à supprimer.

Valeur de retour

Nombre de caractères supprimés.

© 2000-2024, MetaQuotes Ltd.

TrimRight

Supprime toutes les occurences d'un ensemble de caractères spécifiés (et ' ','\t','\r','\n') à la fin de la chaîne de caractères.

```
int  TrimRight(
   const string  targets      // Ensemble de caractères à supprimer
   )
```

Paramètres

targets

[in] Ensemble de caractères à supprimer.

Valeur de retour

Nombre de caractères supprimés.

© 2000-2024, MetaQuotes Ltd.

Clear

Vide la chaîne de caractères.

```
bool  Clear()
```

Valeur de retour

vrai si réalisé avec succès, faux si la chaîne de caractères ne peut pas être vidée.

© 2000-2024, MetaQuotes Ltd.

ToUpper

Transforme une chaîne de caractères en majuscules.

```
bool  ToUpper()
```

Valeur de retour

vrai si réalisé avec succès, faux si la chaîne de caractères n'a pas pu être convertie.

© 2000-2024, MetaQuotes Ltd.

ToLower

Transforme une chaîne de caractères en minuscules.

```
bool  ToLower()
```

Valeur de retour

vrai si réalisé avec succès, faux si la chaîne de caractères n'a pas pu être convertie.

© 2000-2024, MetaQuotes Ltd.

Reverse

Retourne une chaîne de caractères.

```
void  Reverse()
```

© 2000-2024, MetaQuotes Ltd.

Find

Recherche la première occurence d'une chaîne de caractères.

```
int  Find(
   uint          start,          // Position
   const string  substring       // Chaîne de caractères à rechercher
   ) const;
```

Paramètres

start

[in] L'index du caractère de la chaîne correspondant à l'index de départ de la recherche, ou 0 pour commencer au début.

substring

[in] Chaîne de caractères à rechercher.

Valeur de retour

L'index du premier caractère correspondant à la chaîne recherchée, ou -1 si la chaîne de caractères n'a pas été trouvée.

© 2000-2024, MetaQuotes Ltd.

FindRev

Recherche la dernière occurence d'une chaîne de caractères.

```
int  FindRev(
   const string  substring      // Chaîne de caractères à rechercher
   ) const;
```

Paramètres

substring

[in] Chaîne de caractères à rechercher.

Valeur de retour

L'index du dernier caractère correspondant à la chaîne recherchée, ou -1 si la chaîne de caractères n'a pas été trouvée.

© 2000-2024, MetaQuotes Ltd.

Remove

Supprime une sous-chaîne de caractères de la chaîne de caractères.

```
uint  Remove(
   const string  substring      // Sous-chaîne de caractères à supprimer
   )
```

Paramètres

substring

[in] Chaîne de caractères à rechercher.

Valeur de retour

Nombre de sous-chaînes supprimées.

© 2000-2024, MetaQuotes Ltd.

Replace

Remplace une sous-chaîne de la chaîne de caractères.

```
uint  Replace(
   const string  substring,      // Sous-chaîne à remplacer
   const string  newstring       // Nouvelle sous-chaîne
   )
```

Paramètres

substring

[in] Chaîne de caractères à rechercher.

newstring

[in] Sous-chaîne de remplacement.

Valeur de retour

Nombre de sous-chaînes remplacées.

© 2000-2024, MetaQuotes Ltd.

Objets Graphiques

Cette section contient les détails techniques d'utilisation des classes des objets graphiques et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation des classes d'objets graphiques vous fera gagner du temps lors de la création de programmes personnalisés (scripts, experts).

La Bibliothèque Standard MQL5 (en terme d'objets graphiques) est située dans le répertoire de travail du terminal dans le répertoire Include\ChartObjects.

Classe/Groupe	Description
CChartObject	Classe de base d'un objet graphique
Lignes	Groupe de classes "Lignes"
Canaux	Groupe de classes "Canaux"
Outils de Gann	Groupe de classes "Gann"
Outils de Fibonacci	Groupe de classes "Fibonacci"
Outils d'Elliott	Groupe de classes "Elliott"
Formes	Groupe de classes "Formes"
Flèches	Groupe de classes "Flèches"
Contrôles	Groupe de classes "Contrôles"

© 2000-2024, MetaQuotes Ltd.

CChartObject

La classe CChartObject est la classe de base des objets graphique du type de graphique de la bibliothèque MQL5 Standard.

Description

La classe CChartObject fournit un accès simplifié aux fonctions de l'API MQL5 à tous ses descendants.

Déclaration

```
class CChartObject : public CObject
```

Titre

```
#include <ChartObjects\ChartObject.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject

Descendants directs

CChartObjectArrow, CChartObjectBitmap, CChartObjectBmpLabel, CChartObjectCycles, CChartObjectElliottWave3, CChartObjectEllipse, CChartObjectFiboArc, CChartObjectFiboFan, CChartObjectFiboTimes, CChartObjectHLine, CChartObjectRectangle, CChartObjectSubChart, CChartObjectText, CChartObjectTrend, CChartObjectTriangle, CChartObjectVLine

Méthodes de Classe

Attributs	
ChartId	Retourne l'identifiant du graphique contenant un graphique
Window	Retourne le nombre de fenêtres dans lesquelles l'objet graphique est un graphique
Name	Retourne/définit le nom de l'objet graphique
NumPoints	Retourne le nombre de points d'ancrage
Assign	
Attach	Assigne un graphique
SetPoint	Définit le point d'ancrage
Delete	
Delete	Supprime un graphique
Detach	Détache un graphique
Shift	

© 2000-2024, MetaQuotes Ltd.

Attributs	
ShiftObject	Déplacement relatif d'un objet
ShiftPoint	Déplacement relatif du point de l'objet
Propriétés de l'objet	
Time	Retourne/Définit les coordonnées date/heure du point de l'objet
Price	Retourne/Définit les coordonnées prix du point de l'objet
Color	Retourne/Définit la couleur de l'objet
Style	Retourne/Définit le style de ligne de l'objet
Width	Retourne/Définit la largeur de ligne de l'objet
BackGround	Retourne/Définit le flag pour l'affichage de l'arrière plan d'un objet
Selected	Retourne/Définit le flag "selected" de l'objet graphique.
Selectable	Retourne/Définit le flag autorisant ou pas la sélection de l'objet
Description	Retourne/Définit le texte de l'objet
Tooltip	Retourne/Définit l'infobulle de l'objet
Timeframes	Retourne/Définit le masque de visibilité des flags de l'objet
Z_Order	Retourne/Définit la priorité du clic dans un graphique
CreateTime	Retourne l'heure de création d'un objet
Propriétés des niveaux de l'objet	
LevelsCount	Retourne/Définit le nombre de niveaux d'un objet
LevelColor	Retourne/Définit la couleur de ligne du niveau
LevelStyle	Retourne/Définit le style de ligne du niveau
LevelWidth	Retourne/Définit la largeur de ligne du niveau
LevelValue	Retourne/Définit la valeur du niveau
LevelDescription	Retourne/Définit le texte du niveau
Accès aux fonctions de l'API MQL5	
GetInteger	Retourne la valeur des propriétés de l'objet
SetInteger	Définit les propriétés de l'objet
GetDouble	Retourne la valeur des propriétés de l'objet

© 2000-2024, MetaQuotes Ltd.

Attributs	
SetDouble	Définit les propriétés de l'objet
GetString	Retourne la valeur des propriétés de l'objet
SetString	Définit les propriétés de l'objet
Entrée/Sortie	
virtual Save	Méthode virtuelle de sauvegarde d'un fichier
virtual Load	Méthode virtuelle de lecture d'un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

© 2000-2024, MetaQuotes Ltd.

ChartId

Retourne l'identifiant du graphique auquel l'objet graphique appartient.

```
long  ChartId() const
```

Valeur de Retour

Identifiant du Graphique dans lequel se trouve l'objet graphique. Si aucun objet n'est trouvé, retourne -1.

Exemple :

```
//--- exemple d'utilisation de CChartObject::ChartId
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- recupère l'identifiant du Graphique de l'objet
   long chart_id=object.ChartId();
  }
```

© 2000-2024, MetaQuotes Ltd.

Window

Retourne le nombre de fenêtres dans lesquelles l'objet graphique est un graphique

```
int  Window() const
```

Valeur de Retour

Numéro de la fenêtre du graphique où l'objet graphique est positionné (0 - fenêtre principale). Si aucun objet n'est trouvé, retourne -1.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Window
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère la fenêtre de l'objet graphique
   int window=object.Window();
  }
```

© 2000-2024, MetaQuotes Ltd.

Name (Méthode "Get")

Retourne le nom de l'objet graphique.

```
string  Name() const
```

Valeur de Retour

Nom de l'objet graphique lié à l'instance de classe. Si l'objet n'est pas trouvé, retourne NULL.

Name (Méthode "Set")

Définits le nom de l'objet graphique.

```
bool  Name(
   string  name       // nouveau nom
   )
```

Paramètres

name

[in] Le nouveau nom de l'objet graphique.

Valeur de Retour

vrai si réalisé avec succès, faux si le nom ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Name
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- retourne le nom de l'objet graphique
   string object_name=object.Name();
   if(object_name!="MyChartObject")
     {
     //--- définit le nom de l'objet graphique
     object.Name("MyChartObject");
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

NumPoints

Retourne le nombre de points d'ancrage de l'objet graphique.

```
int  NumPoints() const
```

Valeur de Retour

Nombre de points liant un objet graphique lié à une instance de classe. Si aucun objet n'est assigné, retourne 0.

Exemple :

```
//--- exemple d'utilisation de CChartObject::NumPoints
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le nombre de points de l'objet graphique
   int points=object.NumPoints();
  }
```

© 2000-2024, MetaQuotes Ltd.

Attach

Assigne un object graphique à une instance de la classe.

```
bool  Attach(
   long    chart_id,      // Identifiant du graphique
   string  name,          // Nom de l'objet
   int     window,        // Fenêtre du graphique
   int     points         // Nombre de points
   )
```

Paramètres

chart_id

[out] Identifiant du graphique.

name

[in] Nom de l'objet graphique.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

points

[in] Nombre de points d'ancrage de l'objet graphique.

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Attach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- assigne l'objet Graphique
   if(!object.Attach(ChartID(),"MyObject",0,2))
     {
      printf("Erreur d'assignation de l'objet");
      return;
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

SetPoint

Définit les nouvelles coordonnées du point d'ancrage de l'objet graphique.

```
bool  SetPoint(
   int       point,          // Numéro du point
   datetime  new_time,       // Coordoonées en date/heure
   double    new_price       // Coordonnées en prix
   )
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

new_time

[in] Nouvelle valeur de la coordonnée date/heure du point d'ancrage spécifié.

new_price

[in] Nouvelle valeur de la coordonnée prix du point d'ancrage spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si les coordonnées du point ne peuvent pas être modifiées.

Exemple :

```
//--- exemple d'utilisation de CChartObject::SetPoint
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   if(object.NumPoints()>0)
     {
      //--- définit le point de l'objet
      object.SetPoint(0,CurrTime(),price);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime un objet graphique attaché au graphique.

```
bool  Delete()
```

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Delete
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- détache l'objet graphique
   if(!object.Delete())
     {
      printf("Erreur de suppression de l'objet");
      return;
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Detach

Détache l'objet graphique.

```
void  Detach()
```

Valeur de Retour

Aucune.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Detach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- détache l'objet graphique
   object.Detach();
  }
```

© 2000-2024, MetaQuotes Ltd.

ShiftObject

Déplace un objet graphique.

```
bool  ShiftObject(
   datetime  d_time,       // Incrément des coordonnées date/heure
   double    d_price       // Incrément des coordonnées prix
   )
```

Paramètres

d_time

[in] Incrément de la coordonnée date/heure de tous les points d'ancrage.

d_price

[in] Incrément de la coordonnée prix de tous les points d'ancrage.

Valeur de Retour

vrai si réalisé avec succès, faux si l'objet ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CChartObject::ShiftObject
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //--- déplace l'objet graphique
   object.ShiftObject(d_time,d_price);
  }
```

© 2000-2024, MetaQuotes Ltd.

ShiftPoint

Décale un point d'ancrage spécifié de l'objet graphique.

```
bool  ShiftPoint(
   int       point,        // Numéro du point
   datetime  d_time,       // Incrément des coordonnées date/heure
   double    d_price       // Incrément des coordonnées prix
   )
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

d_time

[in] Incrément de la coordonnée date/heure du point spécifié.

d_price

[in] Incrément de la coordonnée prix du point spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si le point ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CChartObject::ShiftPoint
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //---
   if(object.NumPoints()>0)
     {
      //--- décale le point de l'objet graphique
      object.ShiftPoint(0,d_time,d_price);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Time (Méthode "Get")

Retourne la coordonnée date/heure du point d'ancrage spécifié d'un objet graphique.

```
datetime  Time(
   int  point      // Numéro du point
   ) const
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

Valeur de Retour

Coordonnées date/heure du point d'ancrage spécifié de l'objet graphique lié à une instance de classe. Si aucun objet n'est assigné ou si aucun point ne correspond au numéro spécifié, retourne 0.

Time (Méthode "Set")

Retourne la coordonnée date/heure du point d'ancrage spécifié d'un objet graphique.

```
bool  Time(
   int       point,         // Numéro du point
   datetime  new_time       // Date/heure
   )
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

new_time

[in] Nouvelle valeur de la coordonnée date/heure du point d'ancrage spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si la coordonnée ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Time
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- récupère l'heure du point de l'objet
      datetime point_time=object.Time(i);
      if(point_time==0)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- définit la date/heure du point de l'objet
         object.Time(i,TimeCurrent());
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

Price (Méthode "Get")

Retourne la coordonnée prix du point d'ancrage spécifié d'un objet graphique.

```
double  Price(
   int  point      // Numéro du point
   ) const
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

Valeur de Retour

Coordonnées prix du point d'ancrage spécifié de l'objet graphique lié à une instance de classe. Si aucun objet n'est assigné ou si aucun point ne correspond au numéro spécifié, retourne EMPTY_VALUE.

Price (Méthode "Set")

Définit les coordonnées prix du point d'ancrage spécifié de l'objet graphique.

```
bool  Price(
   int     point,          // Numéro du point
   double  new_price       // Prix
   )
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

new_price

[in] Nouvelle valeur de la coordonnée prix du point d'ancrage spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si la coordonnée ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Price
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- récupère le prix du point de l'objet
```

© 2000-2024, MetaQuotes Ltd.

```
      double point_price=object.Price(i);
      if(point_price!=price)
        {
         //--- définit le prix du point de l'objet graphique
         object.Price(i,price);
        }
    }
  }
```

© 2000-2024, MetaQuotes Ltd.

Color (Méthode "Get")

Retourne la couleur de ligne d'un objet graphique.

```
color  Color() const
```

Valeur de Retour

Couleur de ligne de l'objet graphique assigné à l'instance de classe. Si aucun objet n'est assigné, retourne CLR_NONE.

Color (Méthode "Set")

Définit la couleur de ligne de l'objet graphique.

```
bool  Color(
   color  new_color      // Nouvelle couleur
   )
```

Paramètres

new_color

[in] Nouvelle couleur de ligne.

Valeur de Retour

vrai si réalisé avec succès, faux si la couleur ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Color
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--récupère la couleur de l'objet graphique
   color object_color=object.Color();
   if(object_color!=clrRed)
     {
     //--- définit la couleur de l'objet graphique
     object.Color(clrRed);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Style (Méthode "Get")

Retourne le style de ligne d'un objet graphique.

```
ENUM_LINE_STYLE  Style() const
```

Valeur de Retour

Style de ligne de l'objet graphique assigné à l'instance de classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Style (Méthode "Set")

Définit le style de ligne de l'objet graphique.

```
bool  Style(
   ENUM_LINE_STYLE  new_style      // Style
   )
```

Paramètres

new_style

[in] Nouvelle valeur du style de ligne.

Valeur de Retour

vrai si réalisé avec succès, faux si le style ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Style
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le style de ligne de l'objet
   ENUM_LINE_STYLE style=object.Style();
   if(style!=STYLE_SOLID)
     {
      //--- définit le style de ligne de l'objet graphique
      object.Style(STYLE_SOLID);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Width (Méthode "Get")

Retourne la largeur de ligne d'un objet graphique.

```
int  Width() const
```

Valeur de Retour

La largeur de ligne d'un objet graphique attaché à une instance de la classe. Si aucun objet n'est assigné, retourne -1.

Width (Méthode "Set")

Définit la largeur de ligne d'un objet graphique.

```
bool  Width(
   int  new_width      // Epaisseur
   )
```

Paramètres

new_width

[in] Nouvelle valeur d'épaisseur de ligne.

Valeur de Retour

vrai si réalisé avec succès, faux si l'épaisseur de ligne ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Width
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère la largeur de l'objet
   int width=object.Width();
   if(width!=1)
     {
      //--- définit la largeur de l'objet
      object.Width(1);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Background (Méthode "Get")

Retourne le flag pour dessiner un objet graphique en arrière plan.

```
bool  Background() const
```

Valeur de Retour

Flags de dessin d'un objet graphique attaché à une instance de la classe en arrière plan. Si aucun objet n'est attaché, retourne faux.</t3>

Background (Méthode "Set")

Définit le flag pour le dessin d'un objet graphique en arrière plan.

```
bool  Background(
   bool  background      // Valeur du flag
   )
```

Paramètres

background

[in] Nouvelle valeur du flag de dessin de l'objet graphique en arrière plan.

Valeur de Retour

vrai si réalisé avec succès, faux si le flag ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Background
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- recupère le flag d'affichage en arrière plan de l'objet graphique
   bool background_flag=object.Background();
   if(!background_flag)
     {
     //--- définit le flag d'affichage en arrière plan de l'objet graphique
     object.Background(true);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Selected (Méthode "Get")

Retourne le flag indiquant qu'un objet graphique est sélectionné. En d'autres termes, si l'objet graphique est sélectionné ou pas.

```
bool  Selected() const
```

Valeur de Retour

L'état que l'objet, attaché à une instance de la classe, est sélectionné. Si aucun objet n'est assigné, retourne faux.

Selected (Méthode "Set")

Définit le flag indiquant qu'un objet graphique est sélectionné.

```
bool  Selected(
   bool  selected      // Valeur du flag
   )
```

Paramètres

selected

[in] Nouvelle valeur du flag.

Valeur de Retour

vrai si réalisé avec succès, faux si le flag ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Selected
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le flag "selected" de l'objet graphique
   bool selected_flag=object.Selected();
   if(selected_flag)
     {
     //--- définit le flag "selected" de l'objet graphique
     object.Selected(false);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Selectable (Méthode "Get")

Retourne le flag indiquant la possibilité d'un objet graphique à être sélectionné. En d'autres termes, si l'objet graphique peut être sélectionné ou pas.

```
bool  Selectable() const
```

Valeur de Retour

Flag indiquant la possibilité de l'objet, attaché à une instance de classe, à être sélectionné. Si aucun objet n'est assigné, retourne faux.

Selectable (Méthode "Set")

Définit le flag indiquant la possibilité d'un objet graphique à être sélectionné.

```
bool  Selectable(
   bool  selectable      // Valeur du flag
   )
```

Paramètres

selectable

[in] Nouvelle valeur du flag.

Valeur de Retour

vrai si réalisé avec succès, faux si le flag ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Selectable
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le flag "selectable" de l'objet graphique
   bool selectable_flag=object.Selectable();
   if(selectable_flag)
     {
     //--- définit le flag "selectable" de l'objet graphique
     object.Selectable(false);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Description (Méthode "Get")

Retourne une description (texte) d'un objet graphique.

```
string  Description() const
```

Valeur de Retour

Description (texte) de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné, retourne NULL.

Description (Méthode "Set")

Définit la description (texte) d'un objet graphique.

```
bool  Description(
   string  text      // Texte
   )
```

Paramètres

text

[in] Nouvelle description (texte).

Valeur de Retour

vrai si réalisé avec succès, faux si la description ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Description
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère la description de l'objet graphique
   string description=object.Description();
   if(description=="")
     {
      //--- définit la description de l'objet
      object.Description("MyObject");
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Tooltip (Méthode "Get")

Retourne le texte de l'infobulle d'un objet graphique.

```
string  Tooltip() const
```

Valeur de retour

Le texte de l'infobulle d'un objet graphique attaché à une instance de la classe. Si aucun objet n'est assigné, retourne NULL.

Tooltip (Méthode "Set")

Définit le texte de l'infobulle d'un objet graphique.

```
bool  Tooltip(
   string  new_tooltip      // nouveau texte de l'infobulle
   )
```

Paramètres

new_tooltip

[in] Nouveau texte d'une infobulle.

Valeur de retour

vrai si réalisé avec succès, faux si l'infobulle n'a pas pu être modifiée.

Note :

Si la propriété n'est pas définie, alors l'infobulle générée automatiquement par le terminal est affichée. L'infobulle peut être désactivée en utilisant la valeur "\n" (retour à la ligne).

© 2000-2024, MetaQuotes Ltd.

Timeframes (Méthode "Get")

Retourne les flags de visibilité d'un objet graphique.

```
int  Timeframes() const
```

Valeur de Retour

Flags de visibilité d'un objet graphique attaché à une instance de la classe. Si aucun objet n'est assigné, retourne 0.

Timeframes (Méthode "Set")

Définit les flags de visibilité d'un objet graphique.

```
bool  Timeframes(
   int  new_timeframes      // Flags de visibilité
   )
```

Paramètres

new_timeframes

[in] Nouveau flags de visibilité de l'objet graphique.

Valeur de Retour

vrai si réalisé avec succès, faux si les flags de visibilité ne peuvent pas être changés.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Timeframes
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère les espaces de temps de l'objet graphique
   int timeframes=object.Timeframes();
   if(!(timeframes&OBJ_PERIOD_H1))
     {
      //---définit les espaces de temps de l'objet graphique
      object.Timeframes(timeframes|OBJ_PERIOD_H1);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Z_Order (Méthode "Get")

Retourne la priorité de l'objet graphique lors d'un clic sur le graphique (CHARTEVENT_CLICK).

```
long  Z_Order() const
```

Valeur de Retour

Priorité d'un objet graphique assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

Z_Order (Méthode "Set")

Définit la priorité d'un objet graphique lors d'un clic sur le graphique (CHARTEVENT_CLICK).

```
bool  Z_Order(
   long  value      // nouvelle priorité
   )
```

Paramètres

value

[in] Nouvelle priorité d'un objet graphique lors d'un clic sur le graphique (CHARTEVENT_CLICK).

Valeur de Retour

vrai si réalisé avec succès, faux si la priorité ne peut pas être modifiée.

Note

Z_Order est la priorité d'un objet graphique à recevoir les évènements de clic sur un graphique (CHARTEVENT_CLICK). En définissant une valeur supérieure à 0 (valeur par défaut), vous pouvez augmenter la priorité de l'objet.

© 2000-2024, MetaQuotes Ltd.

CreateTime

Retourne la date/heure de création d'un objet graphique.

```
datetime  CreateTime() const
```

Valeur de Retour

Heure de création de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

Exemple :

```
//--- exemple d'utilisation de CChartObject::CreateTime
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère l'heure de création de l'objet graphique
   datetime create_time=object.CreateTime();
  }
```

© 2000-2024, MetaQuotes Ltd.

LevelsCount (Méthode "Get")

Retourne le nombre de niveaux de l'objet graphique.

```
int  LevelsCount() const
```

Valeur de Retour

Nombre de niveaux de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

LevelsCount (Méthode "Set")

Définit le nombre de niveaux de l'objet graphique.

```
bool  LevelsCount(
   int  levels      // Nombre de niveaux
   )
```

Paramètres

levels

[in] Le nouveau nombre de niveaux de l'objet graphique.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre de niveaux ne peut pas être changé.

Exemple :

```
//--- exemple d'utilisation de CChartObject::LevelsCount
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le nombre de niveaux du graphique
   int levels_count=object.LevelsCount();
   //--- ajoute un niveau
   object.LevelsCount(levels_count+1);
  }
```

© 2000-2024, MetaQuotes Ltd.

LevelColor (Méthode "Get")

Retourne la couleur de ligne du niveau spécifié de l'objet graphique.

```
color  LevelColor(
   int  level      // Numéro de niveau
   ) const
```

Paramètres

level

[in] Numéro du niveau.

Valeur de Retour

Couleur de ligne du niveau spécifié de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné ou si aucun niveau ne correspond au numéro spécifié, retourne CLR_NONE.

LevelColor (Méthode "Set")

Définit la couleur de ligne du niveau spécifié de l'objet graphique.

```
bool  LevelColor(
   int    level,          // Numéro de niveau
   color  new_color       // Nouvelle couleur
   )
```

Paramètres

level

[in] Numéro du niveau.

new_color

[in] Nouvelle couleur de ligne du niveau spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si la couleur ne peut pas être modifiée.

Exemple :

```
//--- exemple for CChartObject::LevelColor
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère la couleur du niveau de l'objet graphique
      color level_color=object.LevelColor(i);
      if(level_color!=clrRed)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- modifie la couleur du niveau de l'objet graphique
         object.LevelColor(i,clrRed);
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

LevelStyle (Méthode "Get")

Retourne le style de ligne du niveau spécifié de l'objet graphique.

```
ENUM_LINE_STYLE  LevelStyle(
   int  level      // Numéro de niveau
   ) const
```

Paramètres

level

[in] Numéro du niveau.

Valeur de Retour

Style de ligne du niveau spécifié de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné ou si aucun niveau ne correspond au numéro spécifié, retourne WRONG_VALUE.

LevelStyle (Méthode "Set")

Définit le style de ligne du niveau spécifié de l'objet graphique.

```
int  LevelStyle(
   int              level,      // Numéro du niveau
   ENUM_LINE_STYLE  style       // Style de ligne
   )
```

Paramètres

level

[in] Numéro du niveau.

style

[in] Nouveau style de ligne du niveau spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si le style ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::LevelStyle
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère le style niveau de l'objet graphique
      ENUM_LINE_STYLE level_style=object.LevelStyle(i);
      if(level_style!=STYLE_SOLID)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- définit le style du niveau de l'objet graphique
         object.LevelStyle(i,STYLE_SOLID);
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

LevelWidth (Méthode "Get")

Retourne l'épaisseur de ligne du niveau spécifié de l'objet graphique.

```
int  LevelWidth(
   int  level      // Numéro de niveau
   ) const
```

Paramètres

level

[in] Numéro du niveau.

Valeur de Retour

Epaisseur de ligne du niveau spécifié de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné ou si aucun niveau ne correspond au numéro spécifié, retourne -1.

LevelWidth (Méthode "Set")

Définit l'épaisseur de la ligne du niveau spécifié de l'objet graphique.

```
bool  LevelWidth(
   int  level,          // Numéro de niveau
   int  new_width       // Nouvelle largeur
   )
```

Paramètres

level

[in] Numéro du niveau.

new_width

[in] Nouvelle épaisseur de ligne du niveau spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si l'épaisseur de ligne ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::LevelWidth
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère l'épaisseur du niveau de l'objet graphique
      int level_width=object.LevelWidth(i);
      if(level_width!=1)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- définit l'épaisseur du niveau de l'objet graphique
         object.LevelWidth(i,1);
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

LevelValue (Méthode "Get")

Retourne la valeur du niveau spécifié de l'objet graphique.

```
double  LevelValue(
   int  level      // Numéro de niveau
   ) const
```

Paramètres

level

[in] Numéro du niveau.

Valeur de Retour

La valeur du niveau de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné ou si aucun niveau ne correspond au numéro spécifié, retourne EMPTY_VALUE.

LevelValue (Méthode "Set")

Définit la valeur du niveau de l'objet graphique.

```
bool  LevelValue(
   int     level,          // Numéro du niveau
   double  new_value       // Nouvelle valeur
   )
```

Paramètres

level

[in] Numéro du niveau.

new_value

[in] Nouvelle valeur du niveau spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si la valeur ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::LevelValue
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère la valeur du niveau de l'objet graphique
      double level_value=object.LevelValue(i);
      if(level_value!=0.1*i)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- définit la valeur du niveau de l'objet graphique
         object.LevelValue(i,0.1*i);
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

LevelDescription (Méthode "Get")

Retourne la description du niveau de l'objet graphique.

```
string  LevelDescription(
   int  level       // Numéro du niveau
   ) const
```

Paramètres

level

[in] Numéro du niveau de l'objet graphique.

Valeur de Retour

Description du niveau de l'objet graphique lié à l'instance de class. Si aucun objet n'est assigné ou si aucun niveau ne correspond au numéro spécifié, retourne NULL.

LevelDescription (Méthode "Set")

Définit la description du niveau de l'objet graphique.

```
bool  LevelDescription(
   int     level ,      // Numéro du niveau
   string  text         // Texte
   )
```

Paramètres

level

[in] Numéro du niveau de l'objet graphique.

text

[in] Nouvelle description du niveau de l'objet graphique.

Valeur de Retour

vrai si réalisé avec succès, faux si la description ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::LevelDescription
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère la descrription du niveau de l'objet graphique
      string level_description=object.LevelDescription(i);
      if(level_description=="")
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- modifie la description du niveau de l'objet graphique
         object.LevelDescription(i,"Level_"+IntegerToString(i));
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

GetInteger

Fournit un accès simplifié à la fonction ObjectGetInteger() de l'API MQL5 pour les propriétés de type integer (de type bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, color) liées à une instance d'un objet graphique. Deux signatures existent :

Récupérer la valeur de la propriété sans en vérifier l'exactitude

```
long  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,          // Identifiant de la propriété de ty
   int                           modifier=-1       // Modificateur
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer.

modifier=-1

[in] Modificateur (index) de la propriété de type integer.

Valeur de Retour

En cas de succès, retourne la valeur de la propriété de type integer ; en cas d'erreur, retourne 0.

Récupérer la valeur de la propriété en vérifiant son exactitude

```
bool  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // Identifiant de la propriété de type
   int                           modifier,      // Modificateur
   long&                         value          // Référence à la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer de l'objet.

modifier

[in] Modificateur (index) de la propriété de type integer.

value

[out] Référence à une variable permettant de récupérer la valeur de la propriété.

Valeur de Retour

vrai si réalisé avec succès, faux si la propriété de type integer ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::GetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

© 2000-2024, MetaQuotes Ltd.

```
{
 CChartObject object;
 //--- récupère la couleur de l'objet par la méthode simple
 printf("Objects color is %s",ColorToString(object.GetInteger(OBJPROP_COLOR),true));
 //--- récupère la couleur de l'objet par la méthode classique
 long color_value;
 if(!object.GetInteger(OBJPROP_COLOR,0,color_value))
   {
    printf("Get integer property error %d",GetLastError());
    return;
   }
 else
    printf("Objects color is %s",color_value);
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- récupère les largeurs des lignes de niveaux par la méthode simple
    printf("Level %d width is %d",i,object.GetInteger(OBJPROP_LEVELWIDTH,i));
    //--- récupère les largeurs des lignes de niveaux par la méthode classique
    long width_value;
    if(!object.GetInteger(OBJPROP_LEVELWIDTH,i,width_value))
      {
       printf("Get integer property error %d",GetLastError());
       return;
      }
    else
       printf("Level %d width is %d",i,width_value);
   }
}
```

© 2000-2024, MetaQuotes Ltd.

SetInteger

Fournit un accès simplifié aux fonctions de l'API MQL5 ObjectSetInteger() pour les propriétés de type integer (de type bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, color) liées à une instance d'un objet graphique. Cette fonction peut être appelée de 2 façons :

Définition de la valeur d'une propriété ne nécessitant pas de modificateur

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // Identifiant de la propriété de type
   long                           value         // Valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer de l'objet.

value

[in] Nouvelle valeur pour la propriété de type integer et mutable.

Définition de la valeur d'une propriété en indiquant un modificateur

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // Identifiant de la propriété de type
   int                            modifier,     // Modificateur
   long                           value         // Valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer de l'objet.

modifier

[in] Modificateur (index) de la propriété de type integer.

value

[in] Nouvelle valeur pour la propriété de type integer et mutable.

Valeur de Retour

vrai si réalisé avec succès, faux si la propriété de type integer ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::SetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- définit la nouvelle couleur de l'objet graphique
```

© 2000-2024, MetaQuotes Ltd.

```
  if(!object.SetInteger(OBJPROP_COLOR,clrRed))
    {
     printf("Set integer property error %d",GetLastError());
     return;
    }
  for(int i=0;i<object.LevelsCount();i++)
    {
     //--- définit la largeur des lignes de niveau
     if(!object.SetInteger(OBJPROP_LEVELWIDTH,i,i))
       {
        printf("Set integer property error %d",GetLastError());
        return;
       }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

GetDouble

Fournit un accès simplifié aux fonctions de l'API MQL5 ObjectGetDouble() pour récupérer les valeurs de type double (float ou double) de l'objet graphique assigné à l'instance de classe. Deux signatures existent :

Récupérer la valeur de la propriété sans en vérifier l'exactitude

```
double  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,          // Identifiant de la propriété de typ
   int                          modifier=-1      // Modificateur
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double.

modifier=-1

[in] Modificateur (index) de la propriété de type double.

Valeur de Retour

En cas de succès, retourne la valeur de la propriété de type double ; en cas d'erreur, retourne EMPTY_VALUE.

Récupérer la valeur de la propriété en vérifiant son exactitude

```
bool  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,      // Identifiant de la propriété de type d
   int                          modifier,     // Modificateur
   double&                      value         // Référence vers la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double.

modifier

[in] Modificateur (index) de la propriété de type double.

value

[out] Référence à une variable permettant de récupérer la valeur de la propriété.

Valeur de Retour

vrai si réalisé avec succès, faux si l'objet ne peut pas renvoyer une valeur de type double.

Exemple :

```
//--- exemple d'utilisation de CChartObject::GetDouble
#include <ChartObjects\ChartObject.mqh>
//---
```

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère les valeurs de niveaux par la méthode simple
      printf("Level %d value=%f",i,object.GetDouble(OBJPROP_LEVELVALUE,i));
      //--- get levels value by classic method
      double value;
      if(!object.SetDouble(OBJPROP_LEVELVALUE,i,value))
        {
         printf("Get double property error %d",GetLastError());
         return;
        }
      else
         printf("Level %d value=%f",i,value);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

SetDouble

Fournit un accès simplifié aux fonctions de l'API MQL5 ObjectSetDouble() pour changer les propriétés de type double (type float ou double) lié à une instance de la classe de l'objet graphique. Deux signatures existent :

Définition de la valeur d'une propriété ne nécessitant pas de modificateur

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // Identifiant de propriété de type doub
   double                        value         // Valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double.

value

[in] Nouvelle valeur des propriétés de type double et mutables.

Définition de la valeur d'une propriété en indiquant un modificateur

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,       // Identifiant de la propriété de type
   int                           modifier,      // Modificateur
   double                        value          // Valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double.

modifier

[in] Modificateur (index) de la propriété de type double.

value

[in] Nouvelle valeur de la propriété de type double et mutable.

Valeur de Retour

vrai si réalisé avec succès, faux si la propriété de type double ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::SetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
//---
```

© 2000-2024, MetaQuotes Ltd.

```
  for(int i=0;i<object.LevelsCount();i++)
    {
     //--- définit la valeur du niveau de l'objet graphique
     if(!object.SetDouble(OBJPROP_LEVELVALUE,i,0.1*i))
       {
        printf("Set double property error %d",GetLastError());
        return;
       }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

GetString

Fournit un accès simplifié aux fonctions de l'API MQL5 ObjectGetString() pour récupérer les valeurs de type string de l'objet graphique assigné à l'instance de classe. Deux signatures existent :

Récupérer la valeur de la propriété sans en vérifier l'exactitude

```
string  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,         // Identifiant de la propriété de typ
   int                          modifier=-1      // Modificateur
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type string de l'objet.

modifier=-1

[in] Modificateur (index) de la propriété de type string.

Valeur de Retour

Valeur de la propriété de type string.

Récupérer la valeur de la propriété en vérifiant son exactitude

```
bool  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,      // Identifiant de la propriété de type s
   int                          modifier,     // Modificateur
   string&                      value         // Référence vers la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type string de l'objet.

modifier

[in] Modificateur (index) de la propriété de type string.

value

[out] Référence à une variable permettant de récupérer la valeur de la propriété.

Valeur de Retour

vrai si réalisé avec succès, faux si l'objet ne peut pas renvoyer une valeur de type string.

Exemple :

```
//--- exemple d'utilisation de CChartObject::GetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
  CChartObject object;
  string       value;
  //--- récupère le nom de l'objet par la méthode simple
  printf("Object name is '%s'",object.GetString(OBJPROP_NAME));
  //--- récupère le nom de l'objet par la méthode classique
  if(!object.GetString(OBJPROP_NAME,0,value))
    {
     printf("Get string property error %d",GetLastError());
     return;
    }
  else
     printf("Object name is '%s'",value);
  for(int i=0;i<object.LevelsCount();i++)
    {
     //--- récupère la description des niveaux par la méthode simple
     printf("Level %d description is '%s'",i,object.GetString(OBJPROP_LEVELTEXT,i));
     //--- récupère la description des niveaux par la méthode classique
     if(!object.GetString(OBJPROP_LEVELTEXT,i,value))
       {
        printf("Get string property error %d",GetLastError());
        return;
       }
     else
        printf("Level %d description is '%s'",i,value);
    }
}
```

© 2000-2024, MetaQuotes Ltd.

SetString

Fournit un accès simplifié aux fonctions de l'API MQL5 ObjectSetString() pour changer les propriétés de type string lié à une instance de la classe de l'objet graphique. Deux signatures existent :

Définition de la valeur d'une propriété ne nécessitant pas de modificateur

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // Identifiant de la propriété de type s
   string                        value         // Vaeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type string de l'objet.

value

[in] Nouvelle valeur de la propriété de type string et mutable.

Définition de la valeur d'une propriété en indiquant un modificateur

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,       // Identifiant de la propriété de type
   int                           modifier,      // Modificateur
   string                        value          // Valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type string de l'objet.

modifier

[in] Modificateur (index) de la propriété de type string.

value

[in] Nouvelle valeur de la propriété de type string et mutable.

Valeur de Retour

vrai si réalisé avec succès, faux si la propriété de type string ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::SetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- définit le nouveau nom de l'objet graphique
   if(!object.SetString(OBJPROP_NAME,"MyObject"))
```

© 2000-2024, MetaQuotes Ltd.

```
    {
     printf("Set string property error %d",GetLastError());
     return;
    }
  for(int i=0;i<object.LevelsCount();i++)
    {
     //--- définit la desription des niveaux
     if(!object.SetString(OBJPROP_LEVELTEXT,i,"Level_"+IntegerToString(i)))
       {
        printf("Set string property error %d",GetLastError());
        return;
       }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Save
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CChartObject object=new CChartObject;
   //--- définit les paramètres de l'objet
   //--- . . .
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres de l'objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Load
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CChartObject object;
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilise l'objet
   //--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type d'un objet graphique.

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type de l'objet (0x8888 pour CChartObject).

Exemple :

```
//--- exemple d'utilisation de CChartObject::Type
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le type de l'objet
   int type=object.Type();
  }
```

© 2000-2024, MetaQuotes Ltd.

Objets Lignes

Groupe d'objets graphiques de type "Lignes".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Lignes" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectVLine	Objet graphique "Ligne Verticale"
CChartObjectHLine	Objet graphique "Ligne Horizontale"
CChartObjectTrend	Objet graphique "Ligne de Tendance"
CChartObjectTrendByAngle	Objet graphique "Ligne de Tendance par Angle"
CChartObjectCycles	Objet graphique "Ligne de Cycles"

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

CChartObjectVLine

La classe CChartObjectVLine permet un accès simplifié aux propriétés de l'objet graphique "Ligne Verticale".

Description

La classe CChartObjectVLine fournit un accès aux propriétés de l'objet "Ligne Verticale".

Déclaration

```
class CChartObjectVLine : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectVLine

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Ligne Verticale"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Ligne Verticale".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time           // Coordonnée date/heure
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time

[in] Coordonnée date/heure du point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_VLINE pour CChartObjectVLine).

© 2000-2024, MetaQuotes Ltd.

CChartObjectHLine

La classe CChartObjectHLine permet un accès simplifié aux propriétés de l'objet graphique "Ligne Horizontale".

Description

La classe CChartObjectHLine fournit un accès aux propriétés de l'objet "Ligne Horizontale".

Déclaration

```
class CChartObjectHLine : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Hiérarchie d'héritage

COBject
  CChartObject
    CChartObjectHLine

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Ligne Horizontale"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Ligne Horizontale".

```
bool  Create(
   long    chart_id,      // Identifiant du graphique
   string  name,          // Nom de l'objet
   long    window,        // Fenêtre du graphique
   double  price          // Coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

price

[in] Coordonnée prix du point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_HLINE pour CChartObjectHLine).

© 2000-2024, MetaQuotes Ltd.

CChartObjectTrend

La classe CChartObjectTrend permet un accès simplifié aux propriétés de l'objet graphique "Ligne de Tendance".

Description

La classe CChartObjectTrend fournit un accès aux propriétés de l'objet "Ligne de Tendance".

Déclaration

```
class CChartObjectTrend : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend

Descendants directs

CChartObjectChannel, CChartObjectFibo, CChartObjectFiboChannel, CChartObjectFiboExpansion, CChartObjectGannFan, CChartObjectGannGrid, CChartObjectPitchfork, CChartObjectRegression, CChartObjectStdDevChannel, CChartObjectTrendByAngle

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Ligne de Tendance"
Propriétés	
RayLeft	Retourne/Définit la propriété "Ray Left"
RayRight	Retourne/Définit la propriété "Ray Right"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Ligne de Tendance".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2         // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

RayLeft (Méthode "Get")

Retourne la valeur de la propriété "Ray Left".

```
bool  RayLeft() const
```

Valeur de retour

Valeur de la propriété "Ray Left" assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

RayLeft (Méthode "Set")

Définit la nouvelle valeur de la propriété "Ray Left".

```
bool  RayLeft(
   bool  ray      // flag
   )
```

Paramètres

ray

[in] Nouvelle valeur de la propriété "Ray Left".

Valeur de retour

vrai si réalisé avec succès, faux si le flag n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

RayRight (Méthode "Get")

Retourne la valeur de la propriété "Ray Right".

```
bool  RayRight() const
```

Valeur de retour

Valeur de la propriété "Ray Right" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

RayRight (Méthode "Set")

Définit la nouvelle valeur de la propriété "Ray Right".

```
bool  RayRight(
   bool  ray      // flag
   )
```

Paramètres

ray

[in] Nouvelle valeur de la propriété "Ray Right".

Valeur de retour

vrai si réalisé avec succès, faux si le flag n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier précédemment ouvert en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_TREND pour CChartObjectTrend).

© 2000-2024, MetaQuotes Ltd.

CChartObjectTrendByAngle

La classe CChartObjectTrendByAngle fournit un accès simplifié aux propriétés de l'objet graphique "Ligne de Tendance par Angle".

Description

La classe CChartObjectTrendByAngle fournit un accès aux propriétés de l'objet "Ligne de Tendance par Angle".

Déclaration

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectTrendByAngle

Descendants directs

CChartObjectGannLine

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Ligne de Tendance par Angle"
Propriétés	
Angle	Retourne/Définit la propriété "Angle"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Ligne de Tendance par Angle"

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   long      window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2         // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Angle (Méthode "Get")

Retourne la valeur de la propriété "Angle".

```
double  Angle() const
```

Valeur de retour

Valeur de la propriété "Angle" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

Angle (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Angle".

```
bool  Angle(
   double  angle      // Angle
   )
```

Paramètres

angle

[in] Nouvelle valeur pour la propriété "Angle".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_TRENDBYANGLE pour CChartObjectTrendByAngle).

© 2000-2024, MetaQuotes Ltd.

CChartObjectCycles

La classe CChartObjectCycles permet un accès simplifié aux propriétés de l'objet graphique "Lignes de Cycles" .

Description

La classe CChartObjectCycles fournit un accès aux propriétés de l'objet "Lignes de Cycles".

Déclaration

```
class CChartObjectCycles : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Hiérarchie d'héritage

COBject
  CChartObject
    CChartObjectCycles

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Lignes de Cycles"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Lignes de Cycles"

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   long      window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2         // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_CYCLES pour CChartObjectCycles).

© 2000-2024, MetaQuotes Ltd.

Objets Canaux

Groupe d'objets graphiques de type "Canaux".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Canaux" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectChannel	Objet graphique "Canal Equidistant"
CChartObjectRegression	Objet graphique "Canal de Régression Linéaire"
CChartObjectStdDevChannel	Objet graphique "Canal de Déviation Standard"
CChartObjectPitchfork	Objet graphique "Fourchette d'Andrews"

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

CChartObjectChannel

La classe CChartObjectChannel permet un accès simplifié aux propriétés de l'objet graphique "Canal Equidistant".

Description

La classe CChartObjectChannel fournit un accès aux propriétés de l'objet "Canal Equidistant".

Déclaration

```
class CChartObjectChannel : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectChannel

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Canal Equidistant"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Canal Equidistant"

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Coordonnée date/heure du premier point d'ancrage
   double    price1,        // Coordonnée prix du premier point d'ancrage
   datetime  time2,         // Coordonnée date/heure du deuxième point d'ancrage
   double    price2,        // Coordonnée prix du second point d'ancrage
   datetime  time3,         // Coordonnée date/heure du troisième point d'ancrage
   double    price3         // Coordonnée prix du troisième point d'ancrage
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_CHANNEL pour CChartObjectChannel).

© 2000-2024, MetaQuotes Ltd.

CChartObjectRegression

La classe CChartObjectRegression permet un accès simplifié aux propriétés de l'objet graphique "Canal de Régression Linéaire".

Description

La classe CChartObjectRegression fournit un accès aux propriétés de l'objet "Canal de Régression Linéaire".

Déclaration

```
class CChartObjectRegression : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectRegression

Méthodes de Classe

Création	
Create	Crée un objet graphique "Canal de Régression Linéaire"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique "Canal de Régression Linéaire"

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   long      window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   datetime  time2          // Seconde coordonnée date/heure
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_REGRESSION pour CChartObjectRegression).

© 2000-2024, MetaQuotes Ltd.

CChartObjectStdDevChannel

La class CChartObjectStdDevChannel permet un accès simplifié aux propriétés de l'objet graphique "Canal de Déviation Standard".

Description

La classe CChartObjectStdDevChannel fournit un accès aux propriétés de l'objet "Canal de Déviation Standard".

Déclaration

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectStdDevChannel

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Canal de Déviation Standard"
Propriétés	
Deviation	Retourne/Définit la propriété "Deviation"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Canal de Déviation Standard"

```
bool  Create(
   long      chart_id,       // Identifiant du graphique
   string    name,           // Nom de l'objet
   int       window,         // Fenêtre du graphique
   datetime  time1,          // Première coordonnée date/heure
   datetime  time2,          // Seconde coordonnée date/heure
   double    deviation       // Déviation
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

deviation

[in] Valeur numérique pour la propriété "Deviation".

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Deviation (Méthode "Get")

Retourne la valeur numérique de la propriété "Deviation".

```
double  Deviation() const
```

Valeur de retour

Valeur numérique de la propriété "Deviation", assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

Deviation (Méthode "Set")

Définit la valeur numérique de la propriété "Deviation".

```
bool  Deviation(
   double  deviation      // Déviation
   )
```

Paramètres

deviation

[in] Nouvelle valeur pour la propriété "Deviation".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_STDDEVCHANNEL pour CChartObjectStdDevChannel).

© 2000-2024, MetaQuotes Ltd.

CChartObjectPitchfork

La classe CChartObjectPitchfork permet un accès simplifié aux propriétés de l'objet graphique "Fourchette d'Andrews".

Description

La classe CChartObjectPitchfork fournit un accès aux propriétés de l'objet "Fourchette d'Andrews".

Déclaration

```
class CChartObjectPitchfork : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectPitchfork

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Fourchette d'Andrews"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Fourchette d'Andrews"

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   long      window,        // Fenêtre du graphique
   datetime  time1,         // Coordonnée date/heure du premier point d'ancrage
   double    price1,        // Coordonnée prix du premier point d'ancrage
   datetime  time2,         // Coordonnée date/heure du deuxième point d'ancrage
   double    price2,        // Coordonnée prix du second point d'ancrage
   datetime  time3,         // Coordonnée date/heure du troisième point d'ancrage
   double    price3         // Coordonnée prix du troisième point d'ancrage
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_PITCHFORK pour CChartObjectPitchfork).

© 2000-2024, MetaQuotes Ltd.

Outils de Gann

Groupe d'objets graphiques de type "Outils de Gann".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Outils de Gann" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectGannLine	Objet graphique "Ligne de Gann"
CChartObjectGannFan	Objet graphique "Eventail de Gann"
CChartObjectGannGrid	Objet graphique "Grille de Gann"

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

CChartObjectGannLine

La classe CChartObjectGannLine permet un accès simplifié aux propriétés de l'objet graphique "Ligne de Gann".

Description

La classe CChartObjectGannLine fournit un accès aux propriétés de l'objet "Ligne de Gann".

Déclaration

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

Titre

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectTrendByAngle
        CChartObjectGannLine

Méthodes de Classe

Create	
Create	Crée un objet graphique de type "Ligne de Gann"
Propriétés	
PipsPerBar	Retourne/Définit la propriété "Scale"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Méthodes héritées de la classe CChartObjectTrendByAngle

Angle, Angle, Create

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Ligne de Gann".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    ppb            // Pips par barre
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

ppb

[in] Pips par barre.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

PipsPerBar (Méthode "Get")

Retourne la valeur de la propriété "Pips per bar".

```
double  PipsPerBar() const
```

Valeur de retour

Valeur de la propriété "Pips per bar" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

PipsPerBar (Méthode "Set")

Définit la nouvelle valeur de la propriété "Pips per bar".

```
bool  PipsPerBar(
   double  ppb      // Pips par barre
   )
```

Paramètres

ppb

[in] Nouvelle valeur de la propriété "Pips per bar".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_GANNLINE pour CChartObjectGannLine).

© 2000-2024, MetaQuotes Ltd.

CChartObjectGannFan

La classe CChartObjectGannFan permet un accès simplifié aux propriétés de l'objet graphique "Eventail de Gann".

Description

La classe CChartObjectGannFan fournit un accès aux propriétés de l'objet "Eventail de Gann".

Déclaration

```
class CChartObjectGannFan : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannFan

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Eventail de Gann"
Propriétés	
PipsPerBar	Retourne/Définit la propriété "Pips per bar"
Downtrend	Retourne/Définit la propriété "Downtrend"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Eventail de Gann".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    ppb            // Pips par barre
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

ppb

[in] Pips par barre.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

PipsPerBar (Méthode "Get")

Retourne la valeur de la propriété "Pips per bar".

```
double  PipsPerBar() const
```

Valeur de retour

Valeur de la propriété "Pips per bar" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

PipsPerBar (Méthode "Set")

Définit la nouvelle valeur de la propriété "Pips per bar".

```
bool  PipsPerBar(
   double  ppb      // Pips par barre
   )
```

Paramètres

ppb

[in] Nouvelle valeur de la propriété "Pips per bar".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Downtrend (Méthode "Get")

Retourne la valeur de la propriété "Downtrend".

```
bool  Downtrend() const
```

Valeur de retour

Valeur de la propriété "Downtrend" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Downtrend (Méthode "Set")

Définit la nouvelle valeur de la propriété "Downtrend".

```
bool  Downtrend(
   bool  downtrend      // Valeur du flag
   )
```

Paramètres

downtrend

[in] Nouvelle valeur de la propriété "Downtrend".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_GANNFAN pour CChartObjectGannFan).

© 2000-2024, MetaQuotes Ltd.

CChartObjectGannGrid

La classe CChartObjectGannGrid permet un accès simplifié aux propriétés de l'objet graphique "Grille de Gann".

Description

La classe CChartObjectGannGrid fournit un accès aux propriétés de l'objet "Grille de Gann".

Déclaration

```
class CChartObjectGannGrid : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Hiérarchie d'héritage

COBject
  CChartObject
    CChartObjectTrend
      CChartObjectGannGrid

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Grille de Gann"
Propriétés	
PipsPerBar	Retourne/Définit la propriété "Pips per bar"
Downtrend	Retourne/Définit la propriété "Downtrend"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Grille de Gann".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    ppb            // Pips par barre
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

ppb

[in] Pips par barre.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

PipsPerBar (Méthode "Get")

Retourne la valeur de la propriété "Pips per bar".

```
double  PipsPerBar() const
```

Valeur de retour

Valeur de la propriété "Pips per bar" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

PipsPerBar (Méthode "Set")

Définit la nouvelle valeur de la propriété "Pips per bar".

```
bool  PipsPerBar(
   double  ppb      // Pips par barre
   )
```

Paramètres

ppb

[in] Nouvelle valeur de la propriété "Pips per bar".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Downtrend (Méthode "Get")

Retourne la valeur de la propriété "Downtrend".

```
bool  Downtrend() const
```

Valeur de retour

Valeur de la propriété "Downtrend" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Downtrend (Méthode "Set")

Définit la nouvelle valeur de la propriété "Downtrend".

```
bool  Downtrend(
   bool  downtrend      // Valeur du flag
   )
```

Paramètres

downtrend

[in] Nouvelle valeur de la propriété "Downtrend".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_GANNGRID pour CChartObjectGannGrid).

© 2000-2024, MetaQuotes Ltd.

Outils de Fibonacci

Groupe d'objets graphiques de type "Outils de Fibonacci".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Outils de Fibonacci" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectFibo	Objet graphique "Retracement de Fibonacci"
CChartObjectFiboTimes	Objet graphique "Zones de Temps de Fibonacci"
CChartObjectFiboFan	Objet graphique "Eventail de Fibonacci"
CChartObjectFiboArc	Objet graphique "Arc de Fibonacci"
CChartObjectFiboChannel	Objet graphique "Canal de Fibonacci"
CChartObjectFiboExpansion	Objet graphique "Expansion de Fibonacci"

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

CChartObjectFibo

La classe CChartObjectFibo permet un accès simplifié aux propriétés de l'objet graphique "Retracement de Fibonacci".

Description

La classe CChartObjectFibo fournit un accès aux propriétés de l'objet "Retracement de Fibonacci".

Déclaration

```
class CChartObjectFibo : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFibo

Méthodes de Classe

Create	
Create	Crée un objet graphique de type "Retracement de Fibonacci"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Retracement de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2         // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_FIBO pour CChartObjectFibo).

© 2000-2024, MetaQuotes Ltd.

CChartObjectFiboTimes

La classe CChartObjectFiboTimes permet un accès simplifié aux propriétés de l'objet graphique "Zones de Temps de Fibonacci".

Description

La classe CChartObjectFiboTimes fournit un accès aux propriétés de l'objet "Zones de Temps de Fibonacci".

Déclaration

```
class CChartObjectFiboTimes : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectFiboTimes

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Zones de Temps de Fibonacci"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Zones de Temps de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2         // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_FIBOTIMES pour CChartObjectFiboTimes).

© 2000-2024, MetaQuotes Ltd.

CChartObjectFiboFan

La classe CChartObjectFiboFan permet un accès simplifié aux propriétés de l'objet graphique "Eventail de Fibonacci".

Description

La classe CChartObjectFiboFan fournit un accès aux propriétés de l'objet "Eventail de Fibonacci".

Déclaration

```
class CChartObjectFiboFan : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectFiboFan

Méthodes de Classe

Create	
Create	Crée un objet graphique de type "Eventail de Fibonacci"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Eventail de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2         // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_FIBOFAN pour CChartObjectFiboFan).

© 2000-2024, MetaQuotes Ltd.

CChartObjectFiboArc

La classe CChartObjectFiboArc permet un accès simplifié aux propriétés de l'objet graphique "Arc de Fibonacci".

Description

La classe CChartObjectFiboArc fournit un accès aux propriétés de l'objet "Arc de Fibonacci".

Déclaration

```
class CChartObjectFiboArc : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hiérarchie d'héritage

COBJECT
  CChartObject
    CChartObjectFiboArc

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Arc de Fibonacci"
Propriétés	
Scale	Retourne/Définit la propriété "Scale"
Ellipse	Retourne/Définit la propriété "Ellipse"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Arc de Fibonacci"

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   double    scale          // Echelle
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

scale

[in] Echelle.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Scale (Méthode "Get")

Retourne la valeur de la propriété "Scale".

```
double  Scale() const
```

Valeur de retour

Valeur de la propriété "Scale" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

Scale (Méthode "Set")

Définit une nouvelle valeur à la propriété "Scale".

```
bool  Scale(
   double  scale      // Echelle
   )
```

Paramètres

scale

[in] Nouvelle valeur pour la propriété "Scale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Ellipse (Méthode "Get")

Retourne la valeur de la propriété "Ellipse".

```
bool  Ellipse() const
```

Valeur de retour

Valeur de la propriété "Ellipse" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Ellipse (Méthode "Set")

Définit la nouvelle valeur du flag de la propriété "Ellipse".

```
bool  Ellipse(
   bool  ellipse      // valeur du flag
   )
```

Paramètres

ellipse

[in] Nouvelle valeur pour la propriété "Scale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_FIBOARC pour CChartObjectFiboArc).

© 2000-2024, MetaQuotes Ltd.

CChartObjectFiboChannel

La classe CChartObjectFiboChannel permet un accès simplifié aux propriétés de l'objet graphique "Canal de Fibonacci".

Description

La classe CChartObjectFiboChannel fournit un accès aux propriétés de l'objet "Canal de Fibonacci".

Déclaration

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboChannel

Méthodes de Classe

Create	
Create	Crée un objet graphique de type "Canal de Fibonacci"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Canal de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3         // Troisième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_FIBOCHANNEL pour CChartObjectFiboChannel).

© 2000-2024, MetaQuotes Ltd.

CChartObjectFiboExpansion

La classe CChartObjectFiboExpansion permet un accès simplifié aux propriétés de l'objet graphique "Expansion de Fibonacci".

Description

La classe CChartObjectFiboExpansion fournit un accès aux propriétés de l'objet "Expansion de Fibonacci".

Déclaration

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboExpansion

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Expansion de Fibonacci"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Expansion de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3         // Troisième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_EXPANSION pour CChartObjectFiboExpansion).

© 2000-2024, MetaQuotes Ltd.

Outils d'Elliott

Groupe d'objets graphiques de type "Outils d'Elliott".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Outils d'Elliott" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectElliottWave3	Objet graphique "Vague de Correction"
CChartObjectElliottWave5	Objet graphique "Vague d'Impulsion"

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

CChartObjectElliottWave3

La classe CChartObjectElliottWave3 permet un accès smplifié aux propriétés de l'objet graphique "Vague de Correction".

Description

La classe CChartObjectElliottWave3 fournit un accès aux propriétés de l'objet "Vague de Correction".

Déclaration

```
class CChartObjectElliottWave3 : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectElliottWave3

Descendants directs

CChartObjectElliottWave5

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Vague de Correction"
Propriétés	
Degree	Retourne/Définit la propriété "Degree"
Lignes	Retourne/Définit la propriété "Lines"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Vague de Correction".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3         // Troisième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée date/heure du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Degree (Méthode "Get")

Retourne la valeur de la propriété "Degree".

```
ENUM_ELLIOT_WAVE_DEGREE  Degree() const
```

Valeur de retour

Valeur de la propriété "Degree" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Degree (Méthode "Set")

Définit la nouvelle valeur de la propriété "Degree".

```
bool  Degree(
   ENUM_ELLIOT_WAVE_DEGREE  degree      // valeur de la propriété
   )
```

Paramètres

degree

[in] Nouvelle valeur de la propriété "Degree".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Lines (Méthode "Get")

Retourne la valeur de la propriété "Lines".

```
bool  Lines() const
```

Valeur de retour

Valeur de la propriété "Lines" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Lines (Méthode "Set")

Définit la nouvelle valeur de la propriété "Lines".

```
bool  Lines(
   bool  lines      // valeur du flag
   )
```

Paramètres

lines

[in] Nouvelle valeur de la propriété "Lines".

Valeur de retour

vrai si réalisé avec succès, faux si le flag n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_ELLIOTWAVE3 pour CChartObjectElliottWave3).

© 2000-2024, MetaQuotes Ltd.

CChartObjectElliottWave5

La classe CChartObjectElliottWave5 permet un accès simplifié aux propriété de l'objet graphique "Vague d'Impulsion".

Description

La classe CChartObjectElliottWave5 fournit un accès aux propriétés de l'objet "Vague d'Impulsion".

Déclaration

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

Titre

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectElliottWave3
      CChartObjectElliottWave5

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Vague d'Impulsion"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectElliottWave3

Degree, Degree, Lines, Lines, Create, Save, Load

Voir aussi

© 2000-2024, MetaQuotes Ltd.

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Vague d'Impulsion".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3,        // Troisième coordonnée prix
   datetime  time4,         // Quatrième coordonnée date/heure
   double    price4,        // Quatrième coordonnée prix
   datetime  time5          // Cinquième coordonnée date/heure
   double    price5,        // Cinquième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

time4

[in] Coordonnée date/heure du quatrième point d'ancrage.

© 2000-2024, MetaQuotes Ltd.

price4

[in] Coordonnée prix du quatrième point d'ancrage.

time5

[in] Coordonnée date/heure du cinquième point d'ancrage.

price5

[in] Coordonnée prix du cinquième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_ELLIOTWAVE5 pour CChartObjectElliottWave5).

© 2000-2024, MetaQuotes Ltd.

Objets Formes

Groupe d'objets graphiques de type "Formes".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Formes" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectRectangle	Objet graphique "Rectangle"
CChartObjectTriangle	Objet graphique "Triangle"
CChartObjectEllipse	Objet graphique "Ellipse"

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

CChartObjectRectangle

La classe CChartObjectRectangle permet un accès simplifié aux propriétés de l'objet graphique "Rectangle Rectangle".

Description

La classe CChartObjectRectangle fournit un accès aux propriétés de l'objet "Rectangle".

Déclaration

```
S   class CChartObjectRectangle : public CChartObject
```

Titre

```
    #include <ChartObjects\ChartObjectsShapes.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectRectangle

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Rectangle"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Rectangle".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   long      window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2         // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_RECTANGLE pour CChartObjectRectangle).

© 2000-2024, MetaQuotes Ltd.

CChartObjectTriangle

La classe CChartObjectTriangle permet un accès simplifié aux propriétés de l'objet graphique "Triangle".

Description

La classe CChartObjectTriangle fournit un accès aux propriétés de l'objet "Triangle".

Déclaration

```
class CChartObjectTriangle : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectTriangle

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Triangle"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Triangle".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   long      window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3         // Troisième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_TRIANGLE pour CChartObjectTriangle).

© 2000-2024, MetaQuotes Ltd.

CChartObjectEllipse

La classe CChartObjectEllipse permet un accès simplifié aux propriétés de l'objet graphique "Ellipse".

Description

La classe CChartObjectEllipse fournit un accès aux propriétés de l'objet "Ellipse".

Déclaration

```
class CChartObjectEllipse : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectEllipse

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Ellipse"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Ellipse".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3         // Troisième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_ELLIPSE pour CChartObjectEllipse).

© 2000-2024, MetaQuotes Ltd.

Objets Flèches

Groupe d'objets graphiques Flèches.

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Flèches" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library). Par définition, une flèche est une icone à afficher à l'écran et utilisant un certain code. Il existe 2 types d'objets graphiques de type "Flèches" permettant d'afficher des icones sur les graphiques :

- L'objet "Flèche", vous permettant de spécifier le code de l'icone à afficher.
- Les groupes d'objets pour afficher certains types d'icones (et le code fixé correspondant).

Classe pour l'affichage d'icones avec un code arbitraire

Nom de la classe	Nom de l'objet flèche
CChartObjectArrow	Arrow

Classes pour l'affichage d'icones prédéfinies (code fixé)

Nom de la classe	Nom de l'objet flèche
CChartObjectArrowCheck	Coche
CChartObjectArrowDown	Flèche vers le Haut
CChartObjectArrowUp	Flèche vers le Bas
CChartObjectArrowStop	Signe Stop
CChartObjectArrowThumbDown	Pouce vers le Haut
CChartObjectArrowThumbUp	Pouce vers le Bas
CChartObjectArrowLeftPrice	Etiquette Prix à Gauche
CChartObjectArrowRightPrice	Etiquette Prix à Droite

Voir aussi

Types des objets, Points d'ancrage des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

CChartObjectArrow

La classe CChartObjectArrow permet un accès simplifié aux propriétés de l'objet graphique 'Flèche".

Description

La classe CChartObjectArrow fournit un accès aux propriétés générales des objets 'Flèches" à tous ses descendants.

Déclaration

```
class CChartObjectArrow : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

Hiérarchie d'héritage

COBject
  CChartObject
    CChartObjectArrow

Descendants directs

CChartObjectArrowCheck, CChartObjectArrowDown, CChartObjectArrowLeftPrice, CChartObjectArrowRightPrice, CChartObjectArrowStop, CChartObjectArrowThumbDown, CChartObjectArrowThumbUp, CChartObjectArrowUp

Méthodes de Classe

Création	
Create	Crée un objet graphique "Flèche"
Propriétés	
ArrowCode	Retourne/Définit la propriété "Arrow Code"
Anchor	Retourne/Définit la propriété "Anchor"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Voir aussi

Types des objets, Méthodes d'accrochage des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Flèche".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Numéro de fenêtre
   datetime  time,          // Date/heure
   double    price,         // Prix
   char      code           // Code de flèche
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Nom de l'objet (doit être unique).

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time

[in] Date/heure.

price

[in] Prix.

code

[in] Code de la "Flèche" (Wingdings).

Valeur de retour

vrai - en cas de succès, faux sinon

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrow::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- erreur de création de la flèche
      printf("Arrow create: Error %d!",GetLastError());
      //---
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
     }
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrowCode (Méthode "Get")

Retourne le code du symbole de l'objet "Flèche".

```
char  ArrowCode() const
```

Valeur de retour

Code du symbole utilisé par l'objet "Flèche", assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

ArrowCode (Méthode "Set")

Définit le code du symbole à utiliser par l'objet "Flèche"

```
bool  ArrowCode(
   char  code      // Valeur du code
   )
```

Paramètres

code

[in] nouveau code du symbole à utiliser par l'objet "Flèche" (Wingdings).

Valeur de retour

vrai si réalisé avec succès, faux si le code du symbole n'a pas été changé.

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrow::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   char              code=181;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,code))
     {
      //--- erreur de création de la flèche
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- vous pouvez maintenant utiliser la Flèche
//--- . . .
//--- récupère le code du symbole de la "Flèche"
   if(arrow.ArrowCode()!=code)
     {
     //--- définit le code de la "Flèche"
```

© 2000-2024, MetaQuotes Ltd.

```
      arrow.ArrowCode(code);
      }
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

Anchor (Méthode "Get")

Retourne le type de point d'ancrage de l'objet "Flèche"

```
ENUM_ARROW_ANCHOR  Anchor() const
```

Valeur de retour

Type d'ancrage de l'objet "Flèche", assigné à l'instance de classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Anchor (Méthode "Set")

Définit le type de point d'ancrage de l'objet "Flèche"

```
bool  Anchor(
   ENUM_ARROW_ANCHOR  anchor      // nouveau type d'accrochage
   )
```

Paramètres

anchor

[in] Nouveau type d'accrochage

Valeur de retour

vrai si réalisé avec succès, faux si le type d'accrochage n'a pas été changé.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Anchor
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- erreur de création de la flèche
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- retourne l'ancrage de la flèche
   if(arrow.Anchor()!=anchor)
     {
     //--- définit l'ancrage de la flèche
     arrow.Anchor(anchor);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai - en cas de succès, faux sinon

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrow::Save
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- erreur de création de la flèche
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai - en cas de succès, faux sinon

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrow::Load
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (par exemple, OBJ_ARROW pour CChartObjectArrow)

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrow::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   //--- récupère le type de flèche
   int type=arrow.Type();
  }
```

© 2000-2024, MetaQuotes Ltd.

Flèches prédéfinies

Les classes "Flèches prédéfinies" permettent un accès simplifié aux propriété des objets graphiques suivants :

Nom de la classe	Nom de l'objet Flèche
CChartObjectArrowCheck	"Coche"
CChartObjectArrowDown	"Flèche vers le Bas"
CChartObjectArrowUp	"Flèche vers le Haut"
CChartObjectArrowStop	"Flèche Stop"
CChartObjectArrowThumbDown	"Correct" ("Pouce vers le haut")
CChartObjectArrowThumbUp	"Mauvais" ("Pouce vers le bas")
CChartObjectArrowLeftPrice	Flèche "Prix à gauche"
CChartObjectArrowRightPrice	Flèche "Prix à droite"

Description

Les classes "Flèches prédéfinies" fournissent l'accès aux propriété de l'objet.

Déclarations

```
class CChartObjectArrowCheck      : public CChartObjectArrow;
class CChartObjectArrowDown       : public CChartObjectArrow;
class CChartObjectArrowUp         : public CChartObjectArrow;
class CChartObjectArrowStop       : public CChartObjectArrow;
class CChartObjectArrowThumbDown  : public CChartObjectArrow;
class CChartObjectArrowThumbUp    : public CChartObjectArrow;
class CChartObjectArrowLeftPrice  : public CChartObjectArrow;
class CChartObjectArrowRightPrice : public CChartObjectArrow;
```

Titre

```
<ChartObjects\ChartObjectsArrows.mqh>
```

Méthodes de Classe

Création	
Create	Crée l'objet graphique spécifié
Propriétés	
ArrowCode	</t7><li8>"Stub" pour la méthode de changement de code
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

© 2000-2024, MetaQuotes Ltd.

Voir aussi

Types des objets, Méthodes d'accrochage des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Flèche prédéfinie".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Numéro de fenêtre
   datetime  time,          // Date/heure
   double    price          // Prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time

[in] Date/heure.

price

[in] Prix.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrowCheck::Create
//--- exemple d'utilisation de CChartObjectArrowDown::Create
//--- exemple d'utilisation de CChartObjectArrowUp::Create
//--- exemple d'utilisation de CChartObjectArrowStop::Create
//--- exemple d'utilisation de CChartObjectArrowThumbDown::Create
//--- exemple d'utilisation de CChartObjectArrowThumbUp::Create
//--- exemple d'utilisation de CChartObjectArrowLeftPrice::Create
//--- exemple d'utilisation de CChartObjectArrowRightPrice::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- par exemple, prenez CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- erreur de création de la flèche
       printf("Arrow create: Error %d!",GetLastError());
       //---
       return;
      }
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrowCode

Interdit le changement du code des flèches.

```
bool  ArrowCode(
   char  code      // valeur du code
   )
```

Paramètres

code

[in] n'importe quelle valeur

Valeur de retour

Toujours faux.

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrowCheck::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowDown::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowUp::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowStop::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowThumbDown::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowThumbUp::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowLeftPrice::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowRightPrice::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- par exemple, prenez CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
     {
      //--- erreur de création de la flèche
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- définit le code de la flèche
   if(!arrow.ArrowCode(181))
     {
      //--- ce n'est pas une erreur
      printf("Arrow code can not be changed");
     }
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_ARROW_CHECK pour CChartObjectArrowCheck, OBJ_ARROW_DOWN pour CChartObjectArrowDown, OBJ_ARROW_UP pour CChartObjectArrowUp, OBJ_ARROW_STOP pour CChartObjectArrowStop, OBJ_ARROW_THUMB_DOWN pour CChartObjectArrowThumbDown, OBJ_ARROW_THUMB_UP pour CChartObjectArrowThumbUp, OBJ_ARROW_LEFT_PRICE pour CChartObjectArrowLeftPrice, OBJ_ARROW_RIGHT_PRICE pour CChartObjectArrowRightPrice).

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrowCheck::Type
//--- exemple d'utilisation de CChartObjectArrowDown::Type
//--- exemple d'utilisation de CChartObjectArrowUp::Type
//--- exemple d'utilisation de CChartObjectArrowStop::Type
//--- exemple d'utilisation de CChartObjectArrowThumbDown::Type
//--- exemple d'utilisation de CChartObjectArrowThumbUp::Type
//--- exemple d'utilisation de CChartObjectArrowLeftPrice::Type
//--- exemple d'utilisation de CChartObjectArrowRightPrice::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- par exemple, prenez CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- récupère le type de flèche
   int type=arrow.Type();
  }
```

© 2000-2024, MetaQuotes Ltd.

Objet Contrôle

Groupe d'objets graphiques de type "Contrôle".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Contrôles" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectText	Objet graphique "Texte"
CChartObjectLabel	Objet graphique "Etiquette de Texte"
CChartObjectEdit	Objet graphique "Zone d'édition"
CChartObjectButton	Objet graphique "Bouton"
CChartObjectSubChart	Objet graphique "Graphique"
CChartObjectBitmap	Objet graphique "Bitmap"
CChartObjectBmpLabel	Objet graphique "Etiquette de Bitmap"
CChartObjectRectLabel	Objet graphique "Etiquette Rectangle"

Voir aussi

Types des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

CChartObjectText

La classe CChartObjectText permet un accès simplifié aux propriétés de l'objet graphique "Texte".

Description

La classe CChartObjectText fournit un accès aux propriétés de l'objet "Texte".

Déclaration

```
class CChartObjectText : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectText

Descendants directs

CChartObjectLabel

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Texte"
Propriétés	
Angle	Retourne/Définit la propriété "Angle"
Font	Retourne/Définit la propriété "Font"
FontSize	Retourne/Définit la propriété "FontSize"
Anchor	Retourne/Définit la propriété "Anchor"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Classes dérivées :

- CChartObjectLabel

Voir aussi

Assigne une chaîne de caractèresTypes des objets, Propriétés d'un objet, Méthodes de lien avec un objet, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Texte".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time,          // Coordonnée date/heure
   double    price          // Coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time

[in] Coordonnée date/heure du point d'ancrage.

price

[in] Coordonnée prix du point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Angle (Méthode "Get")

Retourne la valeur de la propriété "Angle".

```
double  Angle() const
```

Valeur de retour

Valeur de la propriété "Angle" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

Angle (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Angle".

```
bool  Angle(
   double  angle      // nouvelle valeur
   )
```

Paramètres

angle

[in] Nouvelle valeur pour la propriété "Angle".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Font (Méthode "Get")

Retourne la valeur de la propriété "Font".

```
string  Font() const
```

Valeur de retour

Valeur de la propriété "Font" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne "".

Font (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Font".

```
bool  Font(
   string  font      // nouvelle police de caractères
   )
```

Paramètres

font

[in] Nouvelle valeur pour la propriété "Font".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

FontSize (Méthode "Get")

Retourne la valeur de la propriété "FontSize".

```
int FontSize() const
```

Valeur de retour

Valeur de la propriété "FontSize" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

FontSize (Méthode "Set")

Définit une nouvelle valeur pour la propriété "FontSize".

```
bool FontSize(
   int size      // nouvelle taille de la police de caractères
   )
```

Paramètres

size

[in] Nouvelle valeur pour la propriété "Font".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Anchor (Méthode "Get")

Retourne la valeur de la propriété "Anchor".

```
ENUM_ANCHOR_POINT  Anchor() const
```

Valeur de retour

Valeur de la propriété "Anchor" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Anchor (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Anchor".

```
bool  Anchor(
   ENUM_ANCHOR_POINT  anchor      // nouvelle valeur
   )
```

Paramètres

anchor

[in] Nouvelle valeur pour la propriété "Anchor".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_TEXT pour CChartObjectText).

© 2000-2024, MetaQuotes Ltd.

CChartObjectLabel

La classe CChartObjectLabel perme un accès simplifié aux propriété de l'objet graphique "Label".

Description

La classe CChartObjectLabel fournit un accès aux propriétés de l'objet "Label".

Déclaration

```
class CChartObjectLabel : public CChartObjectText
```

Titre

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel

Descendants directs

CChartObjectEdit, CChartObjectRectLabel

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Label"
Propriétés	
X_Distance	Retourne/Définit la propriété "X_Distance"
Y_Distance	Retourne/Définit la propriété "Y_Distance"
X_Size	Retourne/Définit la propriété "X_Size"
Y_Size	Retourne/Définit la propriété "Y_Size"
Corner	Retourne/Définit la propriété "Corner"
Time	"Stub" pour le changement de coordonnées date/heure
Price	"Stub" pour le changement de coordonnées prix
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Voir aussi

Types des objets, Propriétés d'un objet, Angle d'un graphique, Points d'ancrage des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Label".

```
bool  Create(
   long    chart_id,      // Identifiant du graphique
   string  name,          // Nom de l'objet
   int     window,        // Fenêtre du graphique
   int     X,             // Coordonnée X
   int     Y              // Coordonnée Y
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

X

[in] Coordonnée X.

Y

[in] Coordonnée Y.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

X_Distance (Méthode "Get")

Retourne la valeur de la propriété "X_Distance".

```
int  X_Distance() const
```

Valeur de retour

Valeur de la propriété "X_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

X_Distance (Méthode "Set")

Définit une nouvelle valeur pour la propriété "X_Distance".

```
bool  X_Distance(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur pour la propriété "X_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Y_Distance (Méthode "Get")

Retourne la valeur de la propriété "Y_Distance".

```
int  Y_Distance() const
```

Valeur de retour

Valeur de la propriété "Y_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

Y_Distance (Méthode "Set")

Définit la nouvelle valeur de la propriété "Y_Distance".

```
bool  Y_Distance(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur pour la propriété "Y_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

X_Size

Retourne la valeur de la propriété "X_Size".

```
int  X_Size() const
```

Valeur de retour

Valeur de la propriété "X_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

Y_Size

Retourne la valeur de la propriété "Y_Size".

```
int  Y_Size() const
```

Valeur de retour

Valeur de la propriété "Y_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

Corner (Méthode "Get")

Retourne la valeur de la propriété "Corner".

```
ENUM_BASE_CORNER  Corner() const
```

Valeur de retour

Valeur de la propriété "Corner" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Corner (Méthode "Set")

Définit une nouvelle pour la propriété "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // nouvelle valeur
   )
```

Paramètres

corner

[in] Nouvelle valeur pour la propriété "Corner".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Time

Interdit le changement de la coordonnée date/heure.

```
bool  Time(
   datetime  time      // n'importe quelle valeur
   )
```

Paramètres

time

[in] N'importe quelle valeur de type datetime.

Valeur de retour

toujours faux.

© 2000-2024, MetaQuotes Ltd.

Price

Interdit le changement de la coordonnée prix.

```
bool  Price(
   double  price      // n'importe quelle valeur
   )
```

Paramètres

price

[in] N'importe quelle valeur de type double.

Valeur de retour

toujours faux.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant de type de l'objet (OBJ_LABEL pour CChartObjectLabel).

© 2000-2024, MetaQuotes Ltd.

CChartObjectEdit

La classe CChartObjectEdit permet un accès simplifié aux propriétés de l'objet graphique "Zone d'édition".

Description

La classe CChartObjectEdit fournit un accès aux propriétés de l'objet "Zone d'édition".

Déclaration

```
class CChartObjectEdit : public CChartObjectLabel
```

Titre

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectEdit

Descendants directs

CChartObjectButton

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Zone d'édition"
Propriétés	
TextAlign	Retourne/Definit la propriété "TextAlign"
X_Size	Retourne la propriété "X Size"
Y_Size	Retourne la propriété "Y Size"
BackColor	Retourne/Définit la propriété "Background Color"
BorderColor	Retourne/Définit la propriété "Border Color"
ReadOnly	Retourne/Définit la propriété "Read Only"
Angle	Retourne/Définit la propriété "Angle"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier

© 2000-2024, MetaQuotes Ltd.

Création	
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Méthodes héritées de la classe CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Voir aussi

Types des objets, Propriétés d'un objet, Angle d'un graphique, Points d'ancrage des objets, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique "zone d'édition".

```
bool  Create(
   long    chart_id,      // Identifiant du graphique
   string  name,          // Nom de l'objet
   int     window,        // Fenêtre du graphique
   int     X,             // Coordonnée X
   int     Y,             // Coordonnée Y
   int     sizeX,         // Taille X
   int     sizeY          // Taille Y
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

X

[in] Coordonnée X.

Y

[in] Coordonnée Y.

sizeX

[in] Taille X.

sizeY

[in] Taille Y.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

TextAlign (Méthode "Get")

Retourne la valeur de la propriété "TextAlign" (mode d'alignement du texte).

```
ENUM_ALIGN_MODE  TextAlign() const
```

Valeur de retour

Valeur de la propriété "TextAlign" de l'objet assigné à l'instance de la classe.

TextAlign (Méthode "Set")

Définit la valeur de la propriété (mode d'alignement du texte).

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // nouvelle valeur
   )
```

Paramètres

align

[in] Nouvelle valeur de la propriété "TextAlign".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

X_Size

Définit une nouvelle valeur pour la propriété "X_Size".

```
bool  X_Size(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle valeur pour la propriété "X_Size".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Y_Size

Définit une nouvelle valeur pour la propriété "Y_Size".

```
bool  Y_Size(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle valeur pour la propriété "Y_Size".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

BackColor (Méthode "Get")

Retourne la valeur de la propriété "BackColor".

```
color  BackColor() const
```

Valeur de retour

Valeur de la propriété "BackColor" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne CLR_NONE.

BackColor (Méthode "Set")

Définit une nouvelle valeur pour la propriété "BackColor".

```
bool  BackColor(
   color  new_color      // nouvelle couleur d'arrière plan
   )
```

Paramètres

new_color

[in] Nouvelle valeur pour la propriété "BackColor".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

BorderColor (Méthode "Get")

Retourne la valeur de la propriété "Border Color".

```
color  BorderColor() const
```

Valeur de retour

Valeur de la propriété "Border Color" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne CLR_NONE.

BorderColor (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Border Color".

```
bool  BorderColor(
   color  new_color      // nouvelle couleur de bordure
   )
```

Paramètres

new_color

[in] Nouvelle couleur pour la propriété "Border Color".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ReadOnly (Méthode "Get")

Retourne la valeur de la propriété "Read Only".

```
bool ReadOnly() const
```

Valeur de retour

Valeur de la propriété "Read Only" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

ReadOnly (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Read Only".

```
bool ReadOnly(
   const bool flag      // nouvelel valeur
   )
```

Paramètres

flag

[in] Nouvelle valeur pour la propriété "Read Only".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Angle

Interdit le changement de la propriété "Angle".

```
bool  Angle(
   double  angle      // n'importe quelle valeur
   )
```

Paramètres

angle

[in] N'importe quelle valeur de type double.

Valeur de retour

toujours faux.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_EDIT pour CChartObjectEdit).

© 2000-2024, MetaQuotes Ltd.

CChartObjectButton

La classe CChartObjectButton permet un accès simplifié aux propriétés de l'objet graphique "Bouton".

Description

La classe CChartObjectButton fournit un accès aux propriétés de l'objet "Bouton".

Déclaration

```
class CChartObjectButton : public CChartObjectEdit
```

Titre

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectEdit
          CChartObjectButton

Descendants directs

CChartObjectPanel

Méthodes de Classe

Création	
Create	Hérité de la class CChartObjectEdit
Propriétés	
Etat	Retourne/Définit l'état du bouton (Appuyé/Relâché)
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Méthodes héritées de la classe CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Méthodes héritées de la classe CChartObjectEdit

X_Size, Y_Size, BackColor, BackColor, BorderColor, BorderColor, ReadOnly, ReadOnly, TextAlign, TextAlign, Angle, Create

Voir aussi

Types des objets, Propriétés d'un objet, Angle d'un graphique, Méthodes de lien avec un objet, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

State (Méthode "Get")

Retourne la valeur de la propriété "State".

```
bool  State() const
```

Valeur de retour

Valeur de la propriété "State" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

State (Méthode "Set")

Définit une nouvelle valeur pour la propriété "State".

```
bool  State(
   bool  state      // nouvelle valeur d'état
   )
```

Paramètres

X

[in] Nouvelle valeur pour la propriété "State".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_BUTTON pour CChartObjectButton).

© 2000-2024, MetaQuotes Ltd.

CChartObjectSubChart

La classe CChartObjectSubChart permet un accès simplifié aux propriétés de l'objet graphique "Graphique".

Description

La classe CChartObjectSubChart fournit un accès aux propriétés de l'objet "Graphique".

Déclaration

```
class CChartObjectSubChart : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectSubChart.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectSubChart

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Graphique"
Propriétés	
X_Distance	Retourne/Définit la propriété "X_Distance"
Y_Distance	Retourne/Définit la propriété "Y_Distance"
Corner	Retourne/Définit la propriété "Corner"
X_Size	Retourne/Définit la propriété "X_Size"
Y_Size	Retourne/Définit la propriété "Y_Size"
Symbol	Retourne/Définit la propriété "Symbol"
Period	Retourne/Définit la propriété "Period"
Scale	Retourne/Définit la propriété "Scale"
DateScale	Retourne/Définit la propriété "Show date scale"
PriceScale	Retourne/Définit la propriété "Show price scale"
Time	"Stub" pour le changement de coordonnées date/heure
Price	"Stub" pour le changement de coordonnées prix
Entrée/Sortie	

© 2000-2024, MetaQuotes Ltd.

Création	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Voir aussi

Types des objets, Propriétés des objet, Angle du graphique, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Sous-Graphique".

```
bool  Create(
   long    chart_id,      // Identifiant du graphique
   string  name,          // Nom de l'objet
   int     window,        // Fenêtre du graphique
   int     X,             // Coordonnée X
   int     Y,             // Coordonnée Y
   int     sizeX,         // Taille X
   int     sizeY          // Taille Y
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

X

[in] Coordonnée X.

Y

[in] Coordonnée Y.

sizeX

[in] Taille X.

sizeY

[in] Taille Y.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

X_Distance (Méthode "Get")

Retourne la valeur de la propriété "X_Distance".

```
int  X_Distance() const
```

Valeur de retour

Valeur de la propriété "X_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

X_Distance (Méthode "Set")

Définit une nouvelle valeur pour la propriété "X_Distance".

```
bool  X_Distance(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur pour la propriété "X_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Y_Distance (Méthode "Get")

Retourne la valeur de la propriété "Y_Distance".

```
int  Y_Distance() const
```

Valeur de retour

Valeur de la propriété "Y_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

Y_Distance (Méthode "Set")

Définit la nouvelle valeur de la propriété "Y_Distance".

```
bool  Y_Distance(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur pour la propriété "Y_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Corner (Méthode "Get")

Retourne la valeur de la propriété "Corner".

```
ENUM_BASE_CORNER Corner() const
```

Valeur de retour

Valeur de la propriété "Corner" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Corner (Méthode "Set")

Définit une nouvelle pour la propriété "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // nouvelle valeur
   )
```

Paramètres

corner

[in] Nouvelle valeur pour la propriété "Corner".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

X_Size (Méthode "Get")

Retourne la valeur de la propriété "X_Size".

```
int  X_Size() const
```

Valeur de retour

Valeur de la propriété "X_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

X_Size (Méthode "Set")

Définit une nouvelle valeur pour la propriété "X_Size".

```
bool  X_Size(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur pour la propriété "X_Size".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Y_Size (Méthode "Get")

Retourne la valeur de la propriété "Y_Size".

```
int  Y_Size() const
```

Valeur de retour

Valeur de la propriété "Y_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

Y_Size (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Y_Size".

```
bool  Y_Size(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur pour la propriété "Y_Size".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Symbol (Méthode "Get")

Retourne la valeur de la propriété "Symbol".

```
string  Symbol() const
```

Valeur de retour

Valeur de la propriété "Symbol" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne "".

Symbol (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Symbol".

```
bool  Symbol(
   string  symbol      // nouveau symbole
   )
```

Paramètres

symbol

[in] Nouvelle valeur pour la propriété "Symbol".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Period (Méthode "Get")

Retourne la valeur de la propriété "Period".

```
int  Period() const
```

Valeur de retour

Valeur de la propriété "Period" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

Period (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Period".

```
bool  Period(
   int  period      // nouvelle valeur
   )
```

Paramètres

period

[in] Nouvelle valeur pour la propriété "Period".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Scale (Méthode "Get")

Retourne la valeur de la propriété "Scale".

```
double  Scale() const
```

Valeur de retour

Valeur de la propriété "Scale" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

Scale (Méthode "Set")

Définit une nouvelle valeur à la propriété "Scale".

```
bool  Scale(
   double  scale      // nouvelle valeur
   )
```

Paramètres

scale

[in] Nouvelle valeur pour la propriété "Scale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

DateScale (Méthode "Get")

Retourne la valeur de la propriété "DateScale".

```
bool  DateScale() const
```

Valeur de retour

Valeur de la propriété "DateScale" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

DateScale (Méthode "Set")

Définit une nouvelle valeur pour la propriété "DateScale".

```
bool  DateScale(
   bool  scale      // nouvelle valeur
   )
```

Paramètres

scale

[in] Nouvelle valeur pour la propriété "DateScale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

PriceScale (Méthode "Get")

Retourne la valeur de la propriété "PriceScale".

```
bool  PriceScale() const
```

Valeur de retour

Valeur la propriété "PriceScale" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

PriceScale (Méthode "Set")

Définit une nouvelle valeur pour la propriété "PriceScale".

```
bool  PriceScale(
   bool  scale      // nouvelle valeur
   )
```

Paramètres

scale

[in] Nouvelle valeur pour la propriété "PriceScale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Time

Interdit le changement de la coordonnée date/heure.

```
bool  Time(
   datetime  time      // n'importe quelle valeur
   )
```

Paramètres

time

[in] N'importe quelle valeur de type datetime.

Valeur de retour

toujours faux.

© 2000-2024, MetaQuotes Ltd.

Price

Interdit le changement de la coordonnée prix.

```
bool  Price(
   double  price      // n'importe quelle valeur
   )
```

Paramètres

price

[in] N'importe quelle valeur de type double.

Valeur de retour

toujours faux.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_CHART pour CChartObjectSubChart).

© 2000-2024, MetaQuotes Ltd.

CChartObjectBitmap

La classe CChartObjectBitmap permet un accès simplifié aux propriétés de l'objet "Bitmap".

Description

La classe CChartObjectBitmap fournit un accès aux propriétés de l'objet "Bitmap".

Déclaration

```
class CChartObjectBitmap : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectBitmap

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Bitmap"
Propriétés	
BmpFile	Retourne/Définit la propriété "BMP Filename"
X_Offset	Retourne/Définit la propriété "X_Offset"
Y_Offset	Retourne/Définit la propriété "Y_Offset"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Voir aussi

Types des objets, Propriétés de l'objet, Objet graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "Bitmap".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time,          // Coordonnée date/heure
   double    price          // Coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time

[in] Coordonnée date/heure du point d'ancrage.

price

[in] Coordonnée prix du point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

BmpFile (Méthode "Get")

Retourne la valeur de la propriété "BmpFile".

```
string  BmpFile() const
```

Valeur de retour

Valeur de la propriété "BmpFile" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

BmpFile (Méthode "Set")

Définit la nouvelle valeur de la propriété "BmpFile".

```
bool  BmpFile(
   string  name      // nouveau nom de fichier
   )
```

Paramètres

name

[in] Nouvelle valeur de la propriété "BmpFile".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

X_Offset (Méthode "Get")

Retourne la valeur de la propriété "X_Offset" (la coordonnée X du coin supérieur gauche du rectangle visible de l'objet).

```
int  X_Offset() const
```

Valeur de retour

Valeur de la propriété "X_Offset" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

X_Offset (Méthode "Set")

Définit la nouvelle valeur de la propriété "X_Offset" (la coordonnée X du coin supérieur gauche du rectangle visible de l'objet). La valeur est définie en pixels relativement au coin suéprieur gauche de l'image originale.

```
bool  X_Offset(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur de la propriété "X_Offset".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Y_Offset (Méthode "Get")

Retourne la valeur de la propriété "Y_Offset" (la coordonnée Y du coin supérieur gauche du rectangle visible de l'objet).

```
int  Y_Offset() const
```

Valeur de retour

Valeur de la propriété "Y_Offset" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

Y_Offset (Méthode "Set")

Définit la nouvelle valeur de la propriété "Y_Offset" (la coordonnée Y du coin supérieur gauche du rectangle visible de l'objet). La valeur est définie en pixels relativement au coin suéprieur gauche de l'image originale.

```
bool  Y_Offset(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur de la propriété "Y_Offset".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_BITMAP pour CChartObjectBitmap).

© 2000-2024, MetaQuotes Ltd.

CChartObjectBmpLabel

La classe CChartObjectBmpLabel permet un accès simplifié aux propriétés de l'objet graphique "Bitmap Label".

Description

La classe CChartObjectBmpLabel fournit un accès aux propriétés de l'objet "Bitmap Label".

Déclaration

```
class CChartObjectBmpLabel : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectBmpLabel

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "BmpLabel"
Propriétés	
X_Distance	Retourne/Définit la propriété "X_Distance"
Y_Distance	Retourne/Définit la propriété "Y_Distance"
X_Offset	Retourne/Définit la propriété "X_Offset"
Y_Offset	Retourne/Définit la propriété "Y_Offset"
Corner	Retourne/Définit la propriété "Corner"
X_Size	Retourne/Définit la propriété "X_Size"
Y_Size	Retourne/Définit la propriété "Y_Size"
BmpFileOn	Retourne/Définit la propriété "BmpFileOn" utilisée lorsqu'un bouton est appuyé (On)
BmpFileOff	Retourne/Définit la propriété "BmpFileOff" utilisée lorsqu'un bouton est relâché (Off)
Etat	Retourne/Définit la propriété "Button State" (Appuyé/Relâché)
Time	"Stub" pour le changement de coordonnées date/heure
Price	"Stub" pour le changement de coordonnées prix

© 2000-2024, MetaQuotes Ltd.

Création	
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Voir aussi

Types des objets, Propriétés des objet, Angle du graphique, Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "BmpLabel".

```
bool  Create(
   long    chart_id,      // Identifiant du graphique
   string  name,          // Nom de l'objet
   int     window,        // Fenêtre du graphique
   int     X,             // Coordonnée X
   int     Y              // Coordonnée Y
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

X

[in] Coordonnée X.

Y

[in] Coordonnée Y.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

X_Distance (Méthode "Get")

Retourne la valeur de la propriété "X_Distance".

```
int  X_Distance() const
```

Valeur de retour

Valeur de la propriété "X_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

X_Distance (Méthode "Set")

Définit une nouvelle valeur pour la propriété "X_Distance".

```
bool  X_Distance(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur pour la propriété "X_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Y_Distance (Méthode "Get")

Retourne la valeur de la propriété "Y_Distance".

```
int  Y_Distance() const
```

Valeur de retour

Valeur de la propriété "Y_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

Y_Distance (Méthode "Set")

Définit la nouvelle valeur de la propriété "Y_Distance".

```
bool  Y_Distance(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur pour la propriété "Y_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

X_Offset (Méthode "Get")

Retourne la valeur de la propriété "X_Offset" (la coordonnée X du coin supérieur gauche du rectangle visible de l'objet).

```
int  X_Offset() const
```

Valeur de retour

Valeur de la propriété "X_Offset" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

X_Offset (Méthode "Set")

Définit la nouvelle valeur de la propriété "X_Offset" (la coordonnée X du coin supérieur gauche du rectangle visible de l'objet). La valeur est définie en pixels relativement au coin suéprieur gauche de l'image originale.

```
bool  X_Offset(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur de la propriété "X_Offset".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Y_Offset (Méthode "Get")

Retourne la valeur de la propriété "Y_Offset" (la coordonnée Y du coin supérieur gauche du rectangle visible de l'objet).

```
int  Y_Offset() const
```

Valeur de retour

Valeur de la propriété "Y_Offset" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

Y_Offset (Méthode "Set")

Définit la nouvelle valeur de la propriété "Y_Offset" (la coordonnée Y du coin supérieur gauche du rectangle visible de l'objet). La valeur est définie en pixels relativement au coin suéprieur gauche de l'image originale.

```
bool  Y_Offset(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur de la propriété "Y_Offset".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Corner (Méthode "Get")

Retourne la valeur de la propriété "Corner".

```
ENUM_BASE_CORNER Corner() const
```

Valeur de retour

Valeur de la propriété "Corner" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Corner (Méthode "Set")

Définit une nouvelle pour la propriété "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // nouvelle valeur
   )
```

Paramètres

corner

[in] Nouvelle valeur pour la propriété "Corner".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

X_Size

Retourne la valeur de la propriété "X_Size".

```
int  X_Size() const
```

Valeur de retour

Valeur de la propriété "X_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

Y_Size

Retourne la valeur de la propriété "Y_Size".

```
int  Y_Size() const
```

Valeur de retour

Valeur de la propriété "Y_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

BmpFileOn (Méthode "Get")

Retourne la valeur de la propriété "BmpFileOn".

```
string  BmpFileOn() const
```

Valeur de retour

Valeur de la propriété "BmpFileOn" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne "".

BmpFileOn (Méthode "Set")

Définit une nouvelle valeur pour la propriété "BmpFileOn".

```
bool  BmpFileOn(
   string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nouvelle valeur pour la propriété "BmpFileOn".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

BmpFileOff (Méthode "Get")

Retourne la valeur de la propriété "BmpFileOff".

```
string  BmpFileOff() const
```

Valeur de retour

Valeur de la propriété "BmpFileOff" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne "".

BmpFileOff (Méthode "Set")

Définit une nouvelle valeur pour la propriété "BmpFileOff".

```
bool  BmpFileOff(
   string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nouvelle valeur pour la propriété "BmpFileOff".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

State (Méthode "Get")

Retourne la valeur de la propriété "State".

```
bool  State() const
```

Valeur de retour

Valeur de la propriété "State" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

State (Méthode "Set")

Définit une nouvelle valeur pour la propriété "State".

```
bool  State(
   bool  state      // nouvelle valeur d'état
   )
```

Paramètres

state

[in] Nouvelle valeur pour la propriété "State".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Time

Interdit le changement de la coordonnée date/heure.

```
bool  Time(
   datetime  time      // n'importe quelle valeur
   )
```

Paramètres

time

[in] N'importe quelle valeur de type datetime.

Valeur de retour

toujours faux.

© 2000-2024, MetaQuotes Ltd.

Price

Interdit le changement de la coordonnée prix.

```
bool  Price(
   double  price      // n'importe quelle valeur
   )
```

Paramètres

price

[in] N'importe quelle valeur de type double.

Valeur de retour

toujours faux.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_BITMAP_LABEL pour CChartObjectBmpLabel).

© 2000-2024, MetaQuotes Ltd.

CChartObjectRectLabel

La classe CChartObjectRectLabel permet un accès simplifié aux propriétés de l'objet graphique "Label Rectangle".

Description

La classe CChartObjectRectLabel fournit un accès aux propriétés de l'objet "Label Rectangle".

Déclaration

```
class CChartObjectRectLabel : public CChartObjectLabel
```

Titre

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Hiérarchie d'héritage

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectRectLabel

Méthodes de Classe

Create	
Create	Crée un objet graphique "Label Rectangle"
Propriétés	
X_Size	Définit la taille horizontale
Y_Size	Définit la taille verticale
BackColor	Retourne/Définit la couleur d'arrière plan
Angle	Une méthode générée par un stub
BorderType	Retourne/Définit le type de bordure
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Méthodes héritées de la classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Méthodes héritées de la classe CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Voir aussi

Types des objets, Propriétés de l'objet, Objet graphiques

© 2000-2024, MetaQuotes Ltd.

Create

Crée un objet graphique de type "CChartObjectRectLabel".

```
bool  Create(
   long    chart_id,      // Identifiant du graphique
   string  name,          // Nom de l'objet
   int     window,        // Fenpetre du graphique
   int     X,             // Coordonnée X
   int     Y,             // Coordonnée Y
   int     sizeX,         // Taille horizontale
   int     sizeY          // Taille verticale
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

X

[in] Coordonnée X.

Y

[in] Coordonnée Y.

sizeX

[in] Taille horizontale.

sizeY

[in] Taille verticale.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

X_Size

Définit la valeur de la propriété "X_Size".

```
bool  X_Size(
   int  size      // Taille horizontale
   )
```

Paramètres

size

[in] Nouvelle taille horizontale.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

Note

Pour récupérer les valeurs des propriétés "X_Size" et "Y_Size", utiliser les méthodes X_Size et Y_Size de la classe parente CChartObjectLabel.

© 2000-2024, MetaQuotes Ltd.

Y_Size

Définit la valeur de la propriété "Y_Size".

```
bool  Y_Size(
   int  size      // Taille verticale
   )
```

Paramètres

size

[in] Nouvelle taille verticale.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

Note

Pour récupérer les valeurs des propriétés "X_Size" et "Y_Size", utiliser les méthodes X_Size et Y_Size de la classe parente CChartObjectLabel.

© 2000-2024, MetaQuotes Ltd.

BackColor

Retourne la couleur d'arrière plan.

```
color  BackColor() const
```

Valeur de retour

Couleur d'arrière plan de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

BackColor

Définit la couleur d'arrière plan.

```
bool  BackColor(
   color  new_color      // Nouvelle couleur
   )
```

Paramètres

new_color

[in] Nouvelle couleur d'arrière plan.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Angle

Une méthode générée vi aun stub.

```
bool  Angle(
   double  angle      // n'importe quelle valeur
   )
```

Paramètres

angle

[in] N'importe quelle valeur de type double type.

Valeur de retour

Toujours faux.

© 2000-2024, MetaQuotes Ltd.

BorderType

Retourne le type de bordure.

```
int  BorderType() const
```

Valeur de retour

Type de bortdure de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

BorderType

Définit le type de bordure.

```
bool  BorderType(
   int  type      // Type de bordure
   )
```

Paramètres

type

[in] Nouveau type de bordure.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_RECTANGLE_LABEL pour CChartObjectRectangleLabel).

© 2000-2024, MetaQuotes Ltd.

Ressources graphiques personnalisées

Cette section fournit les outils permettant de travailler avec des ressources graphiques personnalisées.

Leur utilisation facilite grandement la personnalisation de l'affichage des graphiques, des dessins et des visualisations de données.

Il existe des classes individuelles pour créer les objets graphiques et les primitives, pour dessiner différents types de diagrammes circulaires et des courbes. Différentes options d'affichage des objets sont implémentées : changement du style et de la couleur des lignes, remplissage, utilisation des séries de données sur le graphique, etc.

Classe	Description
CCanvas	Classe de création simplifiée d'images personnalisées
CChartCanvas	Classe de base pour implémenter les classes prévues pour dessiner les graphiques et leurs éléments
CHistogramChart	Classe pour dessiner des histogrammes
CLineChart	Classe pour dessiner des courbes
CPieChart	Classe pour dessiner des diagrammes circulaires

© 2000-2024, MetaQuotes Ltd.

CCanvas

CCanvas est une classe pour la création simplifiée d'images personnalisées.

Description

CCanvas fournit la création de ressouces graphiques (avec ou sans lien avec un object graphique) et le dessin de primitives graphiques.

Déclaration

```
class CCanvas
```

Titre

```
#include <Canvas\Canvas.mqh>
```

Hiérarchie d'héritage

CCanvas

Descendants directs

CChartCanvas, CFlameCanvas

Méthodes de classe par groupes

Création	
Attach	Attache l'objet OBJ_BITMAP_LABEL à une instance de la classe CCanvas
Create	Crée une ressource graphique non liée à un objet Graphique
CreateBitmap	Crée une ressource graphique liée à un objet Graphique
CreateBitmapLabel	Crée une ressource graphique liée à un objet Graphique
Destroy	Détruit une ressource graphique
Propriétés	
ChartObjectName	Retourne le nom d'un objet lié Graphique
ResourceName	Retourne le nom d'une ressource graphique
Width	Retourne la largeur d'une ressource graphique
Height	Retourne la hauteur d'une ressource graphique
LineStyleSet	Définit le style de la ligne
Mise à jour d'un objet à l'écran	
Update	Affiche les changements à l'écran

© 2000-2024, MetaQuotes Ltd.

Création	
Resize	Redimmensionne une ressource graphique
Vidage/Remplissage avec une couleur	
Erase	Vide ou remplit avec la couleur spécifiée
Accès aux données	
PixelGet	Retourne la couleur du point situé aux coordonnées spécifiées
PixelSet	Définit la couleur du point situé aux coordonnées spécifiées
Primitives de dessin	
LineVertical	Dessine une ligne verticale
LineHorizontal	Dessine une ligne horizontale
Line	Dessine une ligne à main levée
Polyline	Dessine une polyline
Polygon	Dessine un polygone
Rectangle	Dessine un rectangle
Circle	Dessine un cercle
Triangle	Dessine un triangle
Ellipse	Dessine une ellipse
Arc	Dessine un arc d'ellipse
Pie	Dessine un secteur d'ellipse
Primitives de dessin avec remplissage	
FillRectangle	Dessine un rectangle plein
FillCircle	Dessine un cercle plein
FillTriangle	Dessine un triangle plein
FillPolygon	Dessine un polygone plein
FillEllipse	Dessine une ellipse pleine
Fill	Remplit une zone
Primitives de dessin avec antialiasing	
PixelSetAA	Dessine un pixel
LineAA	Dessine une ligne

© 2000-2024, MetaQuotes Ltd.

Création	
PolylineAA	Dessine une polyline
PolygonAA	Dessine un polygone
TriangleAA	Dessine un triangle
CircleAA	Dessine un cercle
EllipseAA	Dessine une ellipse
LineWu	Dessine une ligne
PolylineWu	Dessine une polyline
PolygonWu	Dessine un polygone
TriangleWu	Dessine un triangle
CircleWu	Dessine un cercle
EllipseWu	Dessine une ellipse
LineThick	Dessine le segment d'une ligne à main levée avec la largeur spécifiée et utilisant l'antialiasing.
LineThickVertical	Dessine le segment vertical d'une ligne à main levée avec la largeur spécifiée et utilisant l'antialiasing.
LineThickHorizontal	Dessine le segment horizontal d'une ligne à main levée avec la largeur spécifiée et utilisant l'antialiasing.
PolygonSmooth	Dessine un polygone avec la largeur spécifiée et utilisant 2 algorithmes d'antialiasing
PolygonThick	Dessine un polygone avec la largeur spécifiée et utilisant l'antialiasing
PolylineSmooth	Dessine une polyline avec la largeur spécifiée et utilisant 2 algorithmes d'antialiasing
PolylineThick	Dessine une polyline avec la largeur spécifiée et utilisant l'antialiasing
Texte	
FontSet	Définit les paramètres de la police de caractères
FontNameSet	Définit le nom de la police de caractères
FontSizeSet	Définit la taille de la police de caractères
FontFlagsSet	Définit les flags de la police de caractères
FontAngleSet	Définit l'angle de pente de la police de caractères
FontGet	Retourne les paramètres de la police de caractères
FontNameGet	Retourne le nom de la police de caractères

© 2000-2024, MetaQuotes Ltd.

Création	
FontSizeGet	Retourne la taille de la police de caractères
FontFlagsGet	Retourne les flags de la police de caractères
FontAngleGet	Retourne l'angle de pente de la police de caractères
TextOut	Affiche le texte
TextWidth	Retourne la largeur du texte
TextHeight	Retourne la hauteur du texte
TextSize	Retourne la taille du texte
Transparence	
TransparentLevelSet	Définit le niveau de transparence
Entrée/sortie	
LoadFromFile	Lit une image depuis un fichier BMP

© 2000-2024, MetaQuotes Ltd.

Attach

Récupère la ressource graphique d'un objet OBJ_BITMAP_LABEL et l'attache à une instance de la classe CCanvas.

```
bool  Attach(
   const long         chart_id,                          // identifiant du graphic
   const string       objname,                           // nom de l'objet
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // méthode de traitement
  )
```

Crée une ressource graphique pour un objet OBJ_BITMAP_LABEL et l'attache à une instance de la classe CCanvas.

```
bool  Attach(
   const long         chart_id,                          // identifiant du graphic
   const string       objname,                           // nom de l'objet
   const int          width,                             // largeur de l'image en
   const int          height,                            // hauteur de l'image en
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // méthode de traitement
  )
```

Paramètres

chart_id

[out] Identifiant du graphique.

objname

[in] Nom de l'objet graphique.

width

[in] Largeur de l'image dans la ressource.

height

[in] Hauteur de l'image dans la ressource.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Méthode de traitement du canal alpha. Le canal alpha est ignoré par défaut.

Valeur de Retour

true - en cas de succès, false - en cas d'échec à attacher l'objet.

© 2000-2024, MetaQuotes Ltd.

Arc

Dessine l'arc d'une ellipse inscrit dans un rectangle dont les coins sont situés aux points (x1,y1) et (x2,y2). Les limites de l'arc sont coupées par des lignes depuis le centre de l'ellipse, qui s'étendent jusqu'aux points de coordonnées (x3,y3) et (x4,y4).

```
void  Arc(
   int         x1,      // Coordonnée X du coin supérieur gauche du rectangle
   int         y1,      // Coordonnée Y du coin supérieur gauche du rectangle
   int         x2,      // Coordonnée X du coin inférieur droit du rectangle
   int         y2,      // Coordonnée Y du coin inférieur droit du rectangle
   int         x3,      // Coordonnée X du premier point pour trouver les limites de l
   int         y3,      // Coordonnée Y du premier point pour trouver les limites de l
   int         x4,      // Coordonnée X du deuxième point pour trouver les limites de
   int         y4,      // Coordonnée Y du deuxième point pour trouver les limites de
   const uint  clr      // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du coin supérieur gauche formant le rectangle.

y1

[in] Coordonnée Y du coin supérieur gauche formant le rectangle.

x2

[in] Coordonnée X du coin inférieur droit formant le rectangle.

y2

[in] Coordonnée Y du coin inférieur droit formant le rectangle.

x3

[in] Coordonnée X du premier point, sur lequel une ligne provenant du centre du rectangle est dessinée pour obtenir la limite de l'arc

y3

[in] Coordonnée Y du premier point, sur lequel une ligne provenant du centre du rectangle est dessinée pour obtenir la limite de l'arc

x4

[in] Coordonnée X du deuxième point, sur lequel une ligne provenant du centre du rectangle est dessinée pour obtenir la limite de l'arc

y4

[in] Coordonnée Y du deuxième point, sur lequel une ligne provenant du centre du rectangle est dessinée pour obtenir la limite de l'arc

clr

[in] Couleur au format ARGB. Utiliser la fonction ColorToARGB() pour convertir une variable de type color au format ARGB.

© 2000-2024, MetaQuotes Ltd.

Dessine l'arc d'une ellipse dont le centre est au point (x,y), inscrit dans un rectangle avec des rayons rx et ry. Les limites de l'arc sont recoupées du centre de l'ellipse en utilisant des rayons formés par les angles fi3 et fi4.

```
void  Arc(
   int         x,       // Coordonnée X du centre de l'ellipse
   int         y,       // Coordonnée Y du centre de l'ellipse
   int         rx,      // Rayon de l'ellipse sur l'axe X
   int         ry,      // Rayon de l'ellipse sur l'axe Y
   int         fi3,     // Angle du rayon depuis le centre de l'ellipse définissant la
   int         fi4,     // Angle du rayon depuis le centre de l'ellipse définissant la
   const uint  clr      // couleur
   );
```

Dessine l'arc d'une ellipse dont le centre est au point (x,y), inscrit dans un rectangle avec des rayons rx et ry, et retourne également les coordonnées des limites de l'arc. Les limites de l'arc sont recoupées du centre de l'ellipse en utilisant des rayons formés par les angles fi3 et fi4.

```
void  Arc(
   int         x,       // Coordonnée X du centre de l'ellipse
   int         y,       // Coordonnée Y du centre de l'ellipse
   int         rx,      // Rayon de l'ellipse sur l'axe X
   int         ry,      // Rayon de l'ellipse sur l'axe Y
   int         fi3,     // Angle du rayon depuis le centre de l'ellipse définissant la
   int         fi4,     // Angle du rayon depuis le centre de l'ellipse définissant la
   int&        x3,      // Coordonnée X de la première limite de l'arc
   int&        y3,      // Coordonnée Y de la première limite de l'arc
   int&        x4,      // Coordonnée X de la deuxième limite de l'arc
   int&        y4,      // Coordonnée Y de la deuxième limite de l'arc
   const uint  clr      // couleur
   );
```

Paramètres

x

[in] Coordonnée X du centre de l'ellipse.

y

[in] Coordonnée Y du centre de l'ellipse.

rx

[in] Rayon de l'ellipse sur l'axe X, en pixels.

ry

[in] Rayon de l'ellipse sur l'axe X, en pixels.

fi3

[in] Angle en radians, définissant la première limite de l'arc.

fi4

© 2000-2024, MetaQuotes Ltd.

[in] Angle en radians, définissant la deuxième limite de l'arc.

x3

[out] Variable pour récupérer la coordonnée X de la première limite de l'arc.

y3

[out] Variable pour récupérer la coordonnée Y de la première limite de l'arc.

x4

[out] Variable pour récupérer la coordonnée X de la deuxième limite de l'arc.

y4

[out] Variable pour récupérer la coordonnée Y de la deuxième limite de l'arc.

clr

[in] Couleur au format ARGB. Utiliser la fonction ColorToARGB() pour convertir une variable de type color au format ARGB.

Exemples d'appel aux méthodes de la classe :

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- crée le canvas
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Erreur lors de la création du canvas : ",GetLastError());
     }
//--- efface le canvas
   canvas.Erase(clrWhite);
//--- dessine le rectangle
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- dessine le premier arc
   canvas.Arc(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrRed));
   int x1,y1,x2,y2;
//--- dessine le deuxième arc
   canvas.Arc(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,x1,y1,x2,y2,ColorToARGB(cl
//--- affiche les coordonnées de l'arc
   PrintFormat("Premier point de l'arc à (%G,%G), deuxième point de l'arc à (%G,%G)",x
   canvas.CircleAA(x1,y1,3, ColorToARGB(clrRed));
   canvas.CircleAA(x2,y2,3, ColorToARGB(clrBlue));
//--- affiche le canvas mis à jour
   canvas.Update();
  }
```

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Pie

Dessine un secteur plein d'une ellipse inscrite dans un rectangle dont les coins sont situés aux points (x1,y1) et (x2,y2). Les limites du secteur sont coupées par des lignes depuis le centre de l'ellipse, qui s'étendent jusqu'aux points de coordonnées (x3,y3) et (x4,y4).

```
void  Pie(
   int         x1,      // Coordonnée X du coin supérieur gauche du rectangle
   int         y1,      // Coordonnée Y du coin supérieur gauche du rectangle
   int         x2,      // Coordonnée X du coin inférieur droit du rectangle
   int         y2,      // Coordonnée Y du coin inférieur droit du rectangle
   int         x3,      // Coordonnée X du premier point pour trouver les limites de l
   int         y3,      // Coordonnée Y du premier point pour trouver les limites de l
   int         x4,      // Coordonnée X du deuxième point pour trouver les limites de
   int         y4,      // Coordonnée Y du deuxième point pour trouver les limites de
   const uint  clr,     // Couleur de ligne
   const uint  fill_clr // Couleur de remplissage
   );
```

Paramètres

x1

[in] Coordonnée X du coin supérieur gauche formant le rectangle.

y1

[in] Coordonnée Y du coin supérieur gauche formant le rectangle.

x2

[in] Coordonnée X du coin inférieur droit formant le rectangle.

y2

[in] Coordonnée Y du coin inférieur droit formant le rectangle.

x3

[in] Coordonnée X du premier point, sur lequel une ligne provenant du centre du rectangle est dessinée pour obtenir la limite de l'arc

y3

[in] Coordonnée Y du premier point, sur lequel une ligne provenant du centre du rectangle est dessinée pour obtenir la limite de l'arc

x4

[in] Coordonnée X du deuxième point, sur lequel une ligne provenant du centre du rectangle est dessinée pour obtenir la limite de l'arc

y4

[in] Coordonnée Y du deuxième point, sur lequel une ligne provenant du centre du rectangle est dessinée pour obtenir la limite de l'arc

clr

[in] Couleur de la bordure du secteur au format ARGB.

fill_clr

© 2000-2024, MetaQuotes Ltd.

[in] Couleur de remplissage du secteur au format ARGB. Utiliser la fonction ColorToARGB() pour convertir une variable de type color au format ARGB.

Dessine un secteur plein d'une ellipse dont le centre est au point (x,y), inscrit dans un rectangle avec des rayons rx et ry. Les limites du secteur sont recoupées du centre de l'ellipse en utilisant des rayons formés par les angles fi3 et fi4.

```
void  Pie(
   int         x,       // Coordonnée X du centre de l'ellipse
   int         y,       // Coordonnée Y du centre de l'ellipse
   int         rx,      // Rayon de l'ellipse sur l'axe X
   int         ry,      // Rayon de l'ellipse sur l'axe Y
   int         fi3,     // Angle du rayon depuis le centre de l'ellipse définissant la
   int         fi4,     // Angle du rayon depuis le centre de l'ellipse définissant la
   const uint  clr,     // Couleur de ligne
   const uint  fill_clr // Couleur de remplissage
   );
```

Dessine un secteur plein d'une ellipse dont le centre est au point (x,y), inscrit dans un rectangle avec des rayons rx et ry, et retourne également les coordonnées des limites de l'arc. Les limites du secteur sont recoupées du centre de l'ellipse en utilisant des rayons formés par les angles fi3 et fi4.

```
void  Pie(
   int         x,       // Coordonnée X du centre de l'ellipse
   int         y,       // Coordonnée Y du centre de l'ellipse
   int         rx,      // Rayon de l'ellipse sur l'axe X
   int         ry,      // Rayon de l'ellipse sur l'axe Y
   int         fi3,     // Angle du rayon depuis le centre de l'ellipse définissant la
   int         fi4,     // Angle du rayon depuis le centre de l'ellipse définissant la
   int&        x3,      // Coordonnée X de la première limite de l'arc
   int&        y3,      // Coordonnée Y de la première limite de l'arc
   int&        x4,      // Coordonnée X de la deuxième limite de l'arc
   int&        y4,      // Coordonnée Y de la deuxième limite de l'arc
   const uint  clr,     // Couleur de ligne
   const uint  fill_clr // Couleur de remplissage
   );
```

Paramètres

x

[in] Coordonnée X du centre de l'ellipse.

y

[in] Coordonnée Y du centre de l'ellipse.

rx

[in] Rayon de l'ellipse sur l'axe X, en pixels.

ry

© 2000-2024, MetaQuotes Ltd.

[in] Rayon de l'ellipse sur l'axe X, en pixels.

fi3

[in] Angle en radians, définissant la première limite de l'arc.

fi4

[in] Angle en radians, définissant la deuxième limite de l'arc.

x3

[out] Variable pour récupérer la coordonnée X de la première limite de l'arc.

y3

[out] Variable pour récupérer la coordonnée Y de la première limite de l'arc.

x4

[out] Variable pour récupérer la coordonnée X de la deuxième limite de l'arc.

y4

[out] Variable pour récupérer la coordonnée Y de la deuxième limite de l'arc.

clr

[in] Couleur de la bordure du secteur au format ARGB.

fill_clr

[in] Couleur de remplissage du secteur au format ARGB. Utiliser la fonction ColorToARGB() pour convertir une variable de type color au format ARGB.

Exemples d'appel aux méthodes de la classe :

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- crée le canvas
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Erreur lors de la création du canvas : ",GetLastError());
     }
//--- efface le canvas
   canvas.Erase(clrWhite);
//--- dessine le rectangle
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- dessine la première part
   canvas.Pie(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrBlue),ColorToARGB(c
//--- dessine la deuxième part
   canvas.Pie(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,ColorToARGB(clrGreen),Colo
//--- affiche le canvas mis à jour
```

© 2000-2024, MetaQuotes Ltd.

```
   canvas.Update();
   DebugBreak();
  }
```

© 2000-2024, MetaQuotes Ltd.

FillPolygon

Dessine un polygone plein.

```
void  FillPolygon(
   int&          x,       // Tableau des coordonnées X des points du polygone
   int&          y,       // Tableau des coordonnées Y des points du polygone
   const uint   clr       // couleur
   );
```

Paramètres

x

[in] Tableau des coordonnées X des points du polygone.

y

[in] Tableau des coordonnées Y des points du polygone.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

FillEllipse

Dessine une ellipse inscrite dans un rectangle aux coordonnées spécifiées.

```
void  FillPolygon(
   int         x1,      // Coordonnée X du coin supérieur gauche du rectangle
   int         y1,      // Coordonnée Y du coin supérieur gauche du rectangle
   int         x2,      // Coordonnée X du coin inférieur droit du rectangle
   int         y2,      // Coordonnée Y du coin inférieur droit du rectangle
   const uint  clr       // Couleur de l'ellipse
   );
```

Paramètres

x1

[in] Coordonnée X du coin supérieur gauche formant le rectangle.

y1

[in] Coordonnée Y du coin supérieur gauche formant le rectangle.

x2

[in] Coordonnée X du coin inférieur droit formant le rectangle.

y2

[in] Coordonnée Y du coin inférieur droit formant le rectangle.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

GetDefaultColor

Retourne une couleur prédéfinie par son indice.

```
static uint  GetDefaultColor(
   const uint  i       // indice
   );
```

Paramètres

i

[in] Indice de la couleur.

Valeur de Retour

Couleur.

© 2000-2024, MetaQuotes Ltd.

ChartObjectName

Retourne le nom d'un objet lié à l'objet Graphique (chart).

```
string  ChartObjectName();
```

Valeur de retour

le nom d'un objet lié à l'objet graphique (chart)

© 2000-2024, MetaQuotes Ltd.

Circle

Dessine un cercle

```
void  Circle(
   int         x,        // Coordonnée X
   int         y,        // Coordonnée Y
   int         r,        // rayon
   const uint  clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du centre du cercle.

y

[in] Coordonnée Y du centre du cercle.

r

[in] Rayon du cercle.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

CircleAA

Dessine un cercle en utilsant l'algorithme d'anti-aliasing

```
void  CircleAA(
   const int     x,        // coordonnée X
   const int     y,        // coordonnée Y
   const double  r,        // rayon
   const uint    clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du centre du cercle.

y

[in] Coordonnée Y du centre du cercle.

r

[in] Rayon du cercle.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

CircleWu

Dessine un cercle en utilsant l'algorithme d'anti-aliasing Wu.

```
void  CircleWu(
   const int     x,        // coordonnée X
   const int     y,        // coordonnée Y
   const double  r,        // rayon
   const uint    clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du centre du cercle.

y

[in] Coordonnée Y du centre du cercle.

r

[in] Rayon du cercle.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

Create

Crée une ressource graphique sans la liér à l'objet Graphique (chart).

```
virtual bool  Create(
   const string       name,                                  // nom
   const int          width,                                 // largeur
   const int          height,                                // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

Paramètres

name

[in] Base du nom d'une ressource graphique. Un nom de ressource est généré à la création en ajoutant une chaîne pseudo-aléatoire.

width

[in] Largeur (taille selon l'axe X) en pixels.

height

[in] Hauteur (taille selon l'axe Y) en pixels.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Méthode de traitement de la couleur. Voir la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de traitement des couleurs.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateBitmap

Crée une ressource graphique liée à l'objet Graphique (chart).

1. Crée une ressource graphique dans la fenêtre principale du graphique actuel.

```
bool  CreateBitmap(
   const string       name,                                  // nom
   const datetime     time,                                  // date/heure
   const double       price,                                 // prix
   const int          width,                                 // largeur
   const int          height,                                // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

2. Crée une ressource graphique en utilisant l'identifiant d'un graphique et un numéro de sous-fenêtre.

```
bool  CreateBitmap(
   const long         chart_id,                              // identifiant d'un graphi
   const int          subwin,                                // numéro de la sous-fenêt
   const string       name,                                  // nom
   const datetime     time,                                  // date/heure
   const double       price,                                 // prix
   const int          width,                                 // largeur
   const int          height,                                // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

Paramètres

chart_id

[in] Identifiant d'un graphique pour créer un objet.

subwin

[in] Numéro de la sous-fenêtre d'un graphique pour créer un objet.

name

[in] Nom de l'objet Graphique (chart) et base de nom pour une ressource graphique.

time

[in] Coordonnée dans le temps du point d'ancrage de l'objet.

price

[in] Coordonnée en valeur du point d'ancrage de l'objet.

width

[in] Largeur de la ressource graphique (taille selon l'axe X) en pixels.

height

[in] Hauteur de la ressource graphique (taille selong l'axe Y) en pixels.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2024, MetaQuotes Ltd.

[in] Méthode de traitement de la couleur. Voir la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de traitement des couleurs.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si la première fonction est utilisée, l'objet est créé dans la fenêtre principale du graphique courant.

La taille d'un objet correspond à la taille d'une ressource graphique.

© 2000-2024, MetaQuotes Ltd.

CreateBitmapLabel

Crée une ressource graphique liée à l'objet Graphique (chart).

1. Crée une ressource graphique dans la fenêtre principale du graphique actuel.

```
bool  CreateBitmapLabel(
   const string       name,                                  // nom
   const int          x,                                     // coordonnée X
   const int          y,                                     // coordonnée Y
   const int          width,                                 // largeur
   const int          height,                                // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

2. Crée une ressource graphique en utilisant l'identifiant d'un graphique et un numéro de sous-fenêtre.

```
bool  CreateBitmapLabel(
   const long         chart_id,                              // identifiant d'un graphi
   const int          subwin,                                // numéro de la sous-fenêt
   const string       name,                                  // nom
   const int          x,                                     // coordonnée X
   const int          y,                                     // coordonnée Y
   const int          width,                                 // largeur
   const int          height,                                // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

Paramètres

chart_id

[in] Identifiant d'un graphique pour créer un objet.

subwin

[in] Numéro de la sous-fenêtre d'un graphique pour créer un objet.

name

[in] Nom de l'objet Graphique (chart) et base de nom pour une ressource graphique.

x

[in] Coordonnée X du point d'ancrage de l'objet Graphique.

y

[in] Coordonnée Y du point d'ancrage de l'objet Graphique.

width

[in] Largeur de la ressource graphique (taille selon l'axe X) en pixels.

height

[in] Hauteur de la ressource graphique (taille selong l'axe Y) en pixels.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2024, MetaQuotes Ltd.

[in] Méthode de traitement de la couleur. Voir la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de traitement des couleurs.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si la première fonction est utilisée, l'objet est créé dans la fenêtre principale du graphique courant.

La taille d'un objet correspond à la taille d'une ressource graphique.

© 2000-2024, MetaQuotes Ltd.

Destroy

Détruit une ressource graphique.

```
void  Destroy();
```

Note

Si une ressource graphique a été liée à un obhjet Graphique (chart), ce derner est détruit également.

© 2000-2024, MetaQuotes Ltd.

Ellipse

Dessine une ellipse à partir de 2 points.

```
void  Ellipse(
   int         x1,       // coordonnée X
   int         y1,       // coordonnée Y
   int         x2,       // coordonnée X
   int         y2,       // coordonnée Y
   const uint  clr       // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier point formant le ellipse.

y1

[in] Coordonnée Y du premier point formant le ellipse.

x2

[in] Coordonnée X du second point formant le ellipse.

y2

[in] Coordonnée Y du second point formant le ellipse.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

EllipseAA

Dessine une ellipse à partir de 2 points en utilsant l'algorithme d'anti-aliasing.

```
void  EllipseAA(
   int         x1,       // coordonnée X
   int         y1,       // coordonnée Y
   int         x2,       // coordonnée X
   int         y2,       // coordonnée Y
   const uint  clr       // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier point formant le ellipse.

y1

[in] Coordonnée Y du premier point formant le ellipse.

x2

[in] Coordonnée X du second point formant le ellipse.

y2

[in] Coordonnée Y du second point formant le ellipse.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

EllipseWu

Dessine un rectangle à partir de 2 points en utilsant l'algorithme d'anti-aliasing Wu.

```
void  EllipseWu(
   int         x1,       // coordonnée X
   int         y1,       // coordonnée Y
   int         x2,       // coordonnée X
   int         y2,       // coordonnée Y
   const uint  clr       // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier point formant le ellipse.

y1

[in] Coordonnée Y du premier point formant le ellipse.

x2

[in] Coordonnée X du second point formant le ellipse.

y2

[in] Coordonnée Y du second point formant le ellipse.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

Erase

Vide ou remplis avec la couleur spécifiée.

```
void  Erase(
   const uint  clr=0      // couleur
   );
```

Paramètres

clr=0

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

Fill

Remplis une zone.

```
void  Fill(
   int          x,        // Coordonnée X
   int          y,        // Coordonnée Y
   const uint   clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du point de départ du remplissage.

y

[in] Coordonnée Y du point de départ du remplissage.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

FillCircle

Dessine un cercle plein.

```
void  FillCircle(
   int         x,        // Coordonnée X
   int         y,        // Coordonnée Y
   int         r,        // rayon
   const uint  clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du centre du cercle plein.

y

[in] Coordonnée Y du centre du cercle plein.

r

[in] Rayon du cercle plein.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

FillRectangle

Dessine un rectangle plein.

```
void  FillRectangle(
   int         x1,      // coordonnée X
   int         y1,      // coordonnée Y
   int         x2,      // coordonnée X
   int         y2,      // coordonnée Y
   const uint  clr      // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier point formant le rectangle.

y1

[in] Coordonnée Y du premier point formant le rectangle.

x2

[in] Coordonnée X du second point formant le rectangle.

y2

[in] Coordonnée Y du second point formant le rectangle.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

FillTriangle

Dessine un triangle plein.

```
void  FillTriangle(
   int         x1,      // coordonnée X
   int         y1,      // coordonnée Y
   int         x2,      // coordonnée X
   int         y2,      // coordonnée Y
   int         x3,      // coordonnée X
   int         y3,      // coordonnée Y
   const uint  clr      // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier sommet du triangle.

y1

[in] Coordonnée Y du premier sommet du triangle.

x2

[in] Coordonnée X du second sommet du triangle.

y2

[in] Coordonnée Y du second sommet du triangle.

x3

[in] Coordonnée X du troisième sommet du triangle.

y3

[in] Coordonnée Y du troisième sommet du triangle.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

FontAngleGet

Retourne l'angle de pente de la police de caractères.

```
uint  FontAngleGet();
```

Valeur de retour

angle de pente de la police de caractères

© 2000-2024, MetaQuotes Ltd.

FontAngleSet

Définis l'angle de pente de la police de caractères.

```
bool  FontAngleSet(
   uint  angle      // angle
   );
```

Paramètres

angle

[in] Angle de pente de la police de caractères en dixièmes de degrés.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

FontFlagsGet

Retourne les flags de la police de caractères.

```
uint  FontFlagsGet();
```

Valeur de retour

flags de la police de caractères

© 2000-2024, MetaQuotes Ltd.

FontFlagsSet

Définis les flags de la police de caractères.

```
bool  FontFlagsSet(
   uint  flags      // flags
   );
```

Paramètres

flags

[in] Flags de créations de la police de caractères. Voir la description de la fonction TextSetFont() pour en savoir plus sur les flags.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

FontGet

Retourne les paramètres actuels de la police de caractères.

```
void  FontGet(
   string&  name,       // nom
   int&     size,       // taille
   uint&    flags,      // flags
   uint&    angle       // angle d'inclinaison
   );
```

Paramètres

name

[out] Référence à une variable pour retourner le nom de la police de caractères.

size

[out] Référence à une variable pour retourner la taille de la police de caractères.

flags

[out] Référence à une variable pour retourner les flags de la police de caractères.

angle

[out] Référence à une variable pour retourner l'angle de pente de la police de caractères.

© 2000-2024, MetaQuotes Ltd.

FontNameGet

Retourne le nom de la police de caractères.

```
string  FontNameGet();
```

Valeur de retour

nom de la police de caractères

© 2000-2024, MetaQuotes Ltd.

FontNameSet

Définit le nom de la police de caractères.

```
bool  FontNameSet(
   string  name       // nom
   );
```

Paramètres

name

[in] Nom de la police de caractères. Par exemple, "Arial".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

FontSet

Définit la police de caractères actuelle.

```
bool  FontSet(
   const string  name,         // nom
   const int     size,         // taille
   const uint    flags=0,      // flags
   const uint    angle=0       // angle
   );
```

Paramètres

name

[in] Nom de la police de caractères. Par exemple, "Arial".

size

[in] Taille de la police de caractères. Voir la description de la fonction TextSetFont() pour savoir comment définir la taille.

flags=0

[in] Flags de créations de la police de caractères. Voir la description de la fonction TextSetFont() pour en savoir plus sur les flags.

angle=0

[in] Angle de pente de la police de caractères en dixièmes de degrés.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

FontSizeGet

Retourne la taille de la police de caractères.

```
int  FontSizeGet();
```

Valeur de retour

taille de la police de caractères

© 2000-2024, MetaQuotes Ltd.

FontSizeSet

Définit la taille de la police de caractères.

```
bool  FontSizeSet(
   int  size      // taille
   );
```

Paramètres

size

[in] Taille de la police de caractères. Voir la description de la fonction TextSetFont() pour savoir comment définir la taille.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Height

Retourne la taille d'une ressource graphique.

```
int  Height();
```

Valeur de retour

hauteur d'une ressource graphique

© 2000-2024, MetaQuotes Ltd.

Line

Dessine un segment d'une ligne à main levée.

```
void  Line(
   int         x1,      // coordonnée X
   int         y1,      // coordonnée Y
   int         x2,      // coordonnée X
   int         y2,      // coordonnée Y
   const uint  clr      // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier point du segment.

y1

[in] Coordonnée y du premier point du segment.

x2

[in] Coordonnée X du second point du segment.

y2

[in] Coordonnée Y du second point du segment.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

LineAA

Dessine un segment d'une ligne à main levée en utilisant l'algorithme d'anti-aliasing.

```
void  LineAA(
   const int   x1,                 // coordonnée X
   const int   y1,                 // coordonnée Y
   const int   x2,                 // coordonnée X
   const int   y2,                 // coordonnée Y
   const uint  clr,                // couleur
   const uint  style=UINT_MAX      // style de ligne
   );
```

Paramètres

x1

[in] Coordonnée X du premier point du segment.

y1

[in] Coordonnée y du premier point du segment.

x2

[in] Coordonnée X du second point du segment.

y2

[in] Coordonnée Y du second point du segment.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2024, MetaQuotes Ltd.

LineWu

Dessine un segment d'une ligne à main levée en utilisant l'algorithme d'anti-aliasing Wu.

```
void  LineWu(
   const int   x1,                 // coordonnée X
   const int   y1,                 // coordonnée Y
   const int   x2,                 // coordonnée X
   const int   y2,                 // coordonnée Y
   const uint  clr,                // couleur
   const uint  style=UINT_MAX      // style de ligne
   );
```

Paramètres

x1

[in] Coordonnée X du premier point du segment.

y1

[in] Coordonnée y du premier point du segment.

x2

[in] Coordonnée X du second point du segment.

y2

[in] Coordonnée Y du second point du segment.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2024, MetaQuotes Ltd.

LineHorizontal

Dessine un segment d'une ligne horizontale.

```
void  LineHorizontal(
   int         x1,       // coordonnée X
   int         x2,       // coordonnée X
   int         y,        // Coordonnée Y
   const uint  clr       // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier point du segment.

x2

[in] Coordonnée X du second point du segment.

y

[in] Coordonnée Y du segment.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

LineVertical

Dessine un segment de ligne verticale.

```
void  LineVertical(
   int          x,        // Coordonnée X
   int          y1,       // coordonnée Y
   int          y2,       // coordonnée Y
   const uint   clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du segment.

y1

[in] Coordonnée y du premier point du segment.

y2

[in] Coordonnée Y du second point du segment.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

LineStyleSet

Définit le style de ligne.

```
void  LineStyleSet(
   const uint  style      // style
   );
```

Paramètres

style

[in] Style de ligne.

Note

Le paramètre d'entrée peut avoir n'importe quelle valeur de l'énumération ENUM_LINE_STYLE. Il est également possible de créer un style de ligne personnalisée.

© 2000-2024, MetaQuotes Ltd.

LineThick

Dessine le segment d'une ligne à main levée avec la largeur spécifiée et utilisant l'antialiasing.

```
void  LineThick(
   const int      x1,            // coordonnée X du premier point du segment
   const int      y1,            // coordonnée Y du premier point du segment
   const int      x2,            // coordonnée X du deuxième point du segment
   const int      y2,            // coordonnée Y du deuxième point du segment
   const uint     clr,           // couleur
   const int      size,          // épaisseur de ligne
   const uint     style,         // épaisseur de ligne
   ENUM_LINE_END  end_style      // style de fin de ligne
   )
```

Paramètres

x1

[in] Coordonnée X du premier point du segment.

y1

[in] Coordonnée Y du premier point du segment.

x2

[in] Coordonnée X du deuxième point du segment.

y2

[in] Coordonnée Y du deuxième point du segment.

clr

[in] Couleur au format ARGB.

size

[in] Epaisseur de ligne.

style

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

end_style

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_END

ENUM_LINE_END

ID	Description
LINE_END_ROUND	Les extremités des lignes sont arrondies.
LINE_END_BUTT	Les extremités des lignes sont coupées.

© 2000-2024, MetaQuotes Ltd.

ID	Description
LINE_END_SQUARE	Une ligne se termine avec un rectangle plein.

© 2000-2024, MetaQuotes Ltd.

LineThickVertical

Dessine le segment vertical d'une ligne à main levée avec la largeur spécifiée et utilisant l'antialiasing.

```
void  LineThickVertical(
   const int      x,             // coordonnée X du segment
   const int      y1,            // coordonnée Y du premier point du segment
   const int      y2,            // coordonnée Y du deuxième point du segment
   const uint     clr,           // couleur
   const int      size,          // épaisseur de ligne
   const uint     style,         // épaisseur de ligne
   ENUM_LINE_END  end_style      // style de fin de ligne
   )
```

Paramètres

x

[in] Coordonnée X du segment.

y1

[in] Coordonnée Y du premier point du segment.

y2

[in] Coordonnée Y du deuxième point du segment.

clr

[in] Couleur au format ARGB.

size

[in] Epaisseur de ligne.

style

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

end_style

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_END.

© 2000-2024, MetaQuotes Ltd.

LineThickHorizontal

Dessine un segment horizontal d'une ligne à main levée avec la largeur spécifiée et utilisant l'antialiasing.

```
void  LineThickHorizontal(
   const int      x1,            // coordonnée X du premier point du segment
   const int      x2,            // coordonnée X du deuxième point du segment
   const int      y,             // coordonnée Y du segment
   const uint     clr,           // couleur
   const int      size,          // épaisseur de ligne
   const uint     style,         // épaisseur de ligne
   ENUM_LINE_END  end_style      // style de fin de ligne
   )
```

Paramètres

x1

[in] Coordonnée X du premier point du segment.

x2

[in] Coordonnée X du deuxième point du segment.

y

[in] Coordonnée Y du segment.

clr

[in] Couleur au format ARGB.

size

[in] Epaisseur de ligne.

style

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

end_style

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_END.

© 2000-2024, MetaQuotes Ltd.

LoadFromFile

Lit une image depuis un fichier BMP.

```
bool  LoadFromFile(
   const string  filename      // nom du fichier
   );
```

Paramètres

filename

[in] Nom du fichier (incluant l'extension "BMP").

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

PixelGet

Retourne la couleur du point situé aux coordonnées spécifiées.

```
uint  PixelGet(
   const int  x,      // coordonnée X
   const int  y       // coordonnée Y
   );
```

Paramètres

x

[in] Coordonnée X du point.

y

[in] Coordonnée Y du point.

Valeur de retour

Couleur du point au format ARGB.

© 2000-2024, MetaQuotes Ltd.

PixelSet

Définit la couleur du point situé aux coordonnées spécifiées.

```
void  PixelSet(
   const int   x,        // coordonnée X
   const int   y,        // coordonnée Y
   const uint  clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du point.

y

[in] Coordonnée Y du point.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

PixelSetAA

Dessine un point en utilisant l'algorithme d'anti-aliasing.

```
void  PixelSetAA(
   const double  x,        // coordonnée X
   const double  y,        // coordonnée Y
   const uint    clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du point.

y

[in] Coordonnée Y du point.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

Polygon

Dessine un polygone.

```
void  Polygon(
   int&        x[],     // tableau de coordonnées X
   int&        y[],     // tableau de coordonnées Y
   const uint  clr      // couleur
   );
```

Paramètres

x[]

[in] Tableau de coordonnées X des points d'un polygone.

y[]

[in] Tableau de coordonnées Y des points d'un polygone.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

PolygonAA

Dessine un polygone en utilisant l'algorithme d'anti-aliasing.

```
void  PolygonAA(
   int&        x[],                // tableau de coordonnées X
   int&        y[],                // tableau de coordonnées Y
   const uint  clr,                // couleur
   const uint  style=UINT_MAX      // style de ligne
   );
```

Paramètres

x[]

[in] Tableau de coordonnées X des points d'un polygone.

y[]

[in] Tableau de coordonnées Y des points d'un polygone.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2024, MetaQuotes Ltd.

PolygonWu

Dessine un polygone en utilisant l'algorithme d'anti-aliasing Wu.

```
void  PolygonWu(
   int&        x[],                // tableau de coordonnées X
   int&        y[],                // tableau de coordonnées Y
   const uint  clr,                // couleur
   const uint  style=UINT_MAX      // style de ligne
   );
```

Paramètres

x[]

[in] Tableau de coordonnées X des points d'un polygone.

y[]

[in] Tableau de coordonnées Y des points d'un polygone.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2024, MetaQuotes Ltd.

PolygonThick

Dessine un polygone avec la largeur spécifiée et utilisant l'antialiasing.

```
void  PolygonThick(
   const int&      x[],          // tableau des coordonnées X des points du polygone
   const int&      y[],          // tableau des coordonnées Y des points du polygone
   const uint     clr,           // couleur
   const int      size,          // épaisseur de ligne
   const uint     style,         // épaisseur de ligne
   ENUM_LINE_END  end_style      // style de fin de ligne
   )
```

Paramètres

x[]

[in] Tableau des coordonnées X des points du polygone.

y[]

[in] Tableau des coordonnées Y des points du polygone.

clr

[in] Couleur au format ARGB.

size

[in] Epaisseur de ligne.

style

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

end_style

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_END.

© 2000-2024, MetaQuotes Ltd.

PolygonSmooth

Dessine un polygone avec la largeur spécifiée et utilisant 2 algorithmes d'antialiasing de façon consécutive. Les segments individuels sont d'abord lissés selon les courbes de Bezier. L'algorithme d'antialiasing est ensuite appliqué sur le polygone construit à partir de ces segments pour améliorer la qualité du rendu.

```
void  PolygonSmooth(
   int&             x[],                        // tableau des coordonnées X des poi
   int&             y[],                        // tableau des coordonnées Y des poi
   const uint       clr,                        // couleur
   const int        size,                       // épaisseur de ligne
   ENUM_LINE_STYLE  style=STYLE_SOLID,          // épaisseur de ligne
   ENUM_LINE_END    end_style=LINE_END_ROUND,   // style de fin de ligne
   double           tension=0.5,                // valeur du paramètre d'antialiasin
   double           step=10                     // longueur des lignes d'approximati
   )
```

Paramètres

&x[]

[in] Tableau des coordonnées X des points du polygone.

&y[]

[in] Tableau des coordonnées Y des points du polygone.

clr

[in] Couleur au format ARGB.

size

[in] Epaisseur de ligne.

style=STYLE_SOLID

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

end_style=LINE_END_ROUND

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_END.

tension=0.5

[in] Valeur du paramètre de lissage.

step=10

[in] Longueur des lignes d'approximation.

© 2000-2024, MetaQuotes Ltd.

Polyline

Dessine une polyline.

```
void  Polyline(
   int&        x[],      // tableau de coordonnées X
   int&        y[],      // tableau de coordonnées Y
   const uint  clr       // couleur
   );
```

Paramètres

x[]

[in] Tableau de coordonnées X de la polyline.

y[]

[in] Tableau de coordonnées Y de la polyline.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

PolylineSmooth

Dessine une polyline avec la largeur spécifiée et utilisant 2 algorithmes d'antialiasing de façon consécutive. Les segments individuels sont d'abord lissés selon les courbes de Bezier. L'algorithme d'antialiasing est ensuite appliqué sur la polyline construit à partir de ces segments pour améliorer la qualité du rendu.

```
void  PolylineSmooth(
   const int&       x[],                        // tableau des coordonnées X des poir
   const int&       y[],                        // tableau des coordonnées Y des poir
   const uint       clr,                        // couleur
   const int        size,                       // épaisseur de ligne
   ENUM_LINE_STYLE  style=STYLE_SOLID,          // épaisseur de ligne
   ENUM_LINE_END    end_style=LINE_END_ROUND,   // style de fin de ligne
   double           tension=0.5,                // valeur du paramètre d'antialiasing
   double           step=10                     // pas d'approximation
   )
```

Paramètres

&x[]

[in] Tableau des coordonnées X des points de la polyline.

&y[]

[in] Tableau des coordonnées X des points de la polyline.

clr

[in] Couleur au format ARGB.

size

[in] Epaisseur de ligne.

style=STYLE_SOLID

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

end_style=LINE_END_ROUND

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_END.

tension=0.5

[in] Valeur du paramètre de lissage.

step=10

[in] Le pas d'approximation.

© 2000-2024, MetaQuotes Ltd.

PolylineThick

Dessine une polyline avec la largeur spécifiée et utilisant l'antialiasing.

```
void  PolylineThick(
   const int      &x[],          // tableau des coordonnées X des points de la polylir
   const int      &y[],          // tableau des coordonnées Y des points de la polylir
   const uint     clr,           // couleur
   const int      size,          // épaisseur de ligne
   const uint     style,         // épaisseur de ligne
   ENUM_LINE_END  end_style      // style de fin de ligne
   )
```

Paramètres

&x[]

[in] Tableau des coordonnées X des points de la polyline.

&y[]

[in] Tableau des coordonnées X des points de la polyline.

clr

[in] Couleur au format ARGB.

size

[in] Epaisseur de ligne.

style

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

end_style

[in] Le style de ligne est l'une des valeurs de l'énumération ENUM_LINE_END

© 2000-2024, MetaQuotes Ltd.

PolylineWu

Dessine une polyline en utilisant l'algorithme d'anti-aliasing Wu.

```
void  PolylineAA(
   int&        x[],                // tableau de coordonnées X
   int&        y[],                // tableau de coordonnées Y
   const uint  clr,                // couleur
   const uint  style=UINT_MAX      // style de ligne
   );
```

Paramètres

x[]

[in] Tableau de coordonnées X de la polyline.

y[]

[in] Tableau de coordonnées Y de la polyline.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2024, MetaQuotes Ltd.

PolylineAA

Dessine une polyline en utilisant l'algorithme d'anti-aliasing.

```
void  PolylineAA(
   int&        x[],                // tableau de coordonnées X
   int&        y[],                // tableau de coordonnées Y
   const uint  clr,                // couleur
   const uint  style=UINT_MAX      // style de ligne
   );
```

Paramètres

x[]

[in] Tableau de coordonnées X de la polyline.

y[]

[in] Tableau de coordonnées Y de la polyline.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2024, MetaQuotes Ltd.

Rectangle

Dessine un rectangle à partir de 2 points.

```
void  Rectangle(
   int         x1,       // coordonnée X
   int         y1,       // coordonnée Y
   int         x2,       // coordonnée X
   int         y2,       // coordonnée Y
   const uint  clr       // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier point formant le rectangle.

y1

[in] Coordonnée Y du premier point formant le rectangle.

x2

[in] Coordonnée X du second point formant le rectangle.

y2

[in] Coordonnée Y du second point formant le rectangle.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

Resize

Redimmensionne une ressource graphique.

```
bool  Resize(
   const int  width,       // largeur
   const int  height       // hauteur
   );
```

Paramètres

width

[in] Nouvelle largeur de la ressource graphique.

height

[in] Nouvelle hauteur de la ressource graphique.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Lors du redimmensionnement, l'image précédente n'est pas sauvegardée.

© 2000-2024, MetaQuotes Ltd.

ResourceName

Retourne le nom d'une ressource graphique.

```
string  ResourceName();
```

Valeur de retour

nom d'une ressource graphique

© 2000-2024, MetaQuotes Ltd.

TextHeight

Retourne la hauteur du texte.

```
int  TextHeight(
   const string  text      // texte
   );
```

Paramètres

text

[in] Texte à mesurer.

Valeur de retour

hauteur du texte en pixels

Note

La police de caractères actuelle est utilisée pour mesurer le texte.

© 2000-2024, MetaQuotes Ltd.

TextOut

Affiche le texte.

```
void  TextOut(
   int          x,                // coordonnée X
   int          y,                // coordonnée Y
   string       text,             // texte
   const uint   clr,              // couleur
   uint         alignment=0       // alignement
   );
```

Paramètres

x

[in] Coordonnée X du point d'ancrage du texte.

y

[in] Coordonnée Y du point d'ancrage du texte.

text

[in] Texte à afficher.

clr

[in] Couleur au format ARGB.

alignment=0

[in] Méthode d'ancrage du texte. Voir la description de la fonction TextOut() pour en savoir plus sur les différentes méthodes d'ancrage.

Note

La police de caractères actuelle est utilisée pour afficher le texte.

© 2000-2024, MetaQuotes Ltd.

TextSize

Retourne la taille du texte.

```
void  TextSize(
   const string  text,        // texte
   int&          width,       // largeur
   int&          height       // hauteur
   );
```

Paramètres

text

[in] Texte à mesurer.

width

[out] Référence à une variable pour retourner la largeur du texte.

height

[out] Référence à une variable pour retourner la hauteur du texte.

Note

La police de caractères actuelle est utilisée pour mesurer le texte.

© 2000-2024, MetaQuotes Ltd.

TextWidth

Retourne la largeur de texte.

```
int  TextWidth(
   const string  text      // texte
   );
```

Paramètres

text

[in] Texte à mesurer.

Valeur de retour

hauteur du texte en pixels

Note

La police de caractères actuelle est utilisée pour mesurer le texte.

© 2000-2024, MetaQuotes Ltd.

TransparentLevelSet

Définit le niveau de transparence.

```
void  TransparentLevelSet(
   const uchar  value      // valeur
   );
```

Paramètres

value

[in] Nouvelle valeur du niveau de transparence.

Note

0 signifie transparence complète, tandis que 255 signifie opacité totale.

Définir le niveau de transparence affecte tous les objets précédemment dessinés. Le niveau de transparence spécifié n'affecte pas les créations suivantes.

© 2000-2024, MetaQuotes Ltd.

Triangle

Dessine un triangle.

```
void  Triangle(
   int         x1,      // coordonnée X
   int         y1,      // coordonnée Y
   int         x2,      // coordonnée X
   int         y2,      // coordonnée Y
   int         x3,      // coordonnée X
   int         y3,      // coordonnée Y
   const uint  clr      // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier sommet du triangle.

y1

[in] Coordonnée Y du premier sommet du triangle.

x2

[in] Coordonnée X du second sommet du triangle.

y2

[in] Coordonnée Y du second sommet du triangle.

x3

[in] Coordonnée X du troisième sommet du triangle.

y3

[in] Coordonnée Y du troisième sommet du triangle.

clr

[in] Couleur au format ARGB.

© 2000-2024, MetaQuotes Ltd.

TriangleAA

Dessine un triangle en utilisant l'algorithme d'anti-aliasing.

```
void  TriangleAA(
   const int   x1,                // coordonnée X
   const int   y1,                // coordonnée Y
   const int   x2,                // coordonnée X
   const int   y2,                // coordonnée Y
   const int   x3,                // coordonnée X
   const int   y3,                // coordonnée Y
   const uint  clr,               // couleur
   const uint  style=UINT_MAX     // style de ligne
   );
```

Paramètres

x1

[in] Coordonnée X du premier sommet du triangle.

y1

[in] Coordonnée Y du premier sommet du triangle.

x2

[in] Coordonnée X du second sommet du triangle.

y2

[in] Coordonnée Y du second sommet du triangle.

x3

[in] Coordonnée X du troisième sommet du triangle.

y3

[in] Coordonnée Y du troisième sommet du triangle.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2024, MetaQuotes Ltd.

TriangleWu

Dessine un triangle en utilisant l'algorithme d'anti-aliasing Wu.

```
void  TriangleAA(
   const int   x1,                // coordonnée X
   const int   y1,                // coordonnée Y
   const int   x2,                // coordonnée X
   const int   y2,                // coordonnée Y
   const int   x3,                // coordonnée X
   const int   y3,                // coordonnée Y
   const uint  clr,               // couleur
   const uint  style=UINT_MAX     // style de ligne
   );
```

Paramètres

x1

[in] Coordonnée X du premier sommet du triangle.

y1

[in] Coordonnée Y du premier sommet du triangle.

x2

[in] Coordonnée X du second sommet du triangle.

y2

[in] Coordonnée Y du second sommet du triangle.

x3

[in] Coordonnée X du troisième sommet du triangle.

y3

[in] Coordonnée Y du troisième sommet du triangle.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2024, MetaQuotes Ltd.

Update

Affiche les changements sur l'écran.

```
void  Update(
   const bool  redraw=true      // flag
   );
```

Paramètres

redraw=true

Flag d'un graphique pour demander le rafraîchissement.

© 2000-2024, MetaQuotes Ltd.

Width

Retourne la largeur d'une ressource graphique.

```
int  Width();
```

Valeur de retour

largeur de la ressource graphique

© 2000-2024, MetaQuotes Ltd.

CChartCanvas

Classe de base pour implémenter les classes utilisées pour dessiner les graphiques et leurs éléments.

Description

Cette classe inclut des méthodes pour travailler avec les éléments de base des graphiques : axes de coordonnées et leurs marques, légende du graphique, grille, arrière plan, etc. Vous pouvez ici personnaliser les options d'affichage des éléments : visibilité, couleur du texte, etc.

Déclaration

```
class CChartCanvas : public CCanvas
```

Titre

```
#include <Canvas\Charts\ChartCanvas.mqh>
```

Hiérarchie d'héritage

CCanvas

CChartCanvas

Descendants directs

CHistogramChart, CLineChart, CPieChart

Méthodes de la classe

Méthode	Action
ColorBackground	Retourne et définit la couleur d'arrière plan.
ColorBorder	Retourne et définit la couleur des bordures.
ColorText	Retourne et définit la couleur du texte.
ColorGrid	Retourne et définit la couleur de la grille.
MaxData	Retourne et définit la quantité maximum de données (séries) autorisées.
MaxDescrLen	Retourne et définit la longueur maximum des descripteurs.
ShowFlags	Retourne et définit la visibilité des élements du graphique.
IsShowLegend	Retourne et définit la visibilité de la légende sur le graphique.
IsShowScaleLeft	Retourne la visibilité de l'échelle des valeurs sur la gauche.

© 2000-2024, MetaQuotes Ltd.

Méthode	Action
IsShowScaleRight	Retourne la visibilité de l'échelle des valeurs sur la droite.
IsShowScaleTop	Retourne la visibilité de l'échelle des valeurs en haut.
IsShowScaleBottom	Retourne la visibilité de l'échelle des valeurs en bas.
IsShowGrid	Retourne la visibilité de la grille sur le graphique.
IsShowDescriptors	Retourne la visibilité des descripteurs sur le graphique.
IsShowPercent	Retourne la visibilité des pourcentages sur le graphique.
aVScaleMin	Retourne et définit la valeur minimum sur l'échelle verticales.
VScaleMax	Retourne et définit la valeur maximum sur l'échelle verticale.
NumGrid	Retourne et définit le nombre de divisions de l'échelle verticale lors de l'affichage de la grille sur le graphique.
DataOffset	Retourne et définit la valeur de décalage des données.
DataTotal	Retourne le nombre total de séries de données sur le graphique.
DrawDescriptors	Méthode virtuelle pour dessiner les descripteurs.
DrawData	Méthode virtuelle pour dessiner les séries de données à l'indice spécifié.
Create	Méthode virtuelle pour créer une ressource graphique.
AllowedShowFlags	Définit l'ensemble de flags de visibilité des élements du graphique.
ShowLegend	Définit la visibilité de la légende.
ShowScaleLeft	Définit la visibilité de l'échelle gauche.
ShowScaleRight	Définit la visibilité de l'échelle droite.
ShowScaleTop	Définit la visibilité de l'échelle en haut.
ShowScaleBottom	Définit la visibilité de l'échelle en bas.

© 2000-2024, MetaQuotes Ltd.

Méthode	Action
ShowGrid	Définit la visibilité de la grille.
ShowDescriptors	Définit la visibilité des descripteurs.
ShowValue	Définit la visibilité des valeurs.
ShowPercent	Définit la visibilité des pourcentages.
LegendAlignment	Définit l'alignement du texte de la légende.
Accumulative	Définit le flag d'accumulation des valeurs pour les séries.
VScaleParams	Définit les paramètres de l'échelle verticale des valeurs.
DescriptorUpdate	Met à jour la valeur du descripteur de séries (à la position spécifiée).
ColorUpdate	Met à jour les couleurs des séries (à la position spécifiée).
ValuesCheck	Effectue des calculs internes pour le dessin du graphique.
Redraw	Redesine le graphique.
DrawBackground	Dessine l'arrière plan.
DrawLegend	Redessine la légende.
DrawLegendVertical	Dessine une légende verticale.
DrawLegendHorizontal	Dessine une légende horizontale.
CalcScales	Calcule les coordonnées de l'échelle.
DrawScales	Redessine toutes les échelles de valeurs.
DrawScaleLeft	Redessine l'échelle gauche de valeurs.
DrawScaleRight	Redessine l'échelle droite de valeurs.
DrawScaleTop	Redessine l'échelle de valeurs en haut.
DrawScaleBottom	Redessine l'échelle de valeurs en bas.
DrawGrid	Redesine le graphique.
DrawChart	Redesine le graphique.

Méthodes héritées de la classe CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu,

© 2000-2024, MetaQuotes Ltd.

TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

© 2000-2024, MetaQuotes Ltd.

ColorBackground (méthode Get)

Retourne la couleur d'arrière plan.

```
uint  ColorBackground()
```

Valeur de Retour

Couleur d'arrière plan.

ColorBackground (méthode Set)

Définit la couleur d'arrière plan.

```
void  ColorBackground(
  const uint  value,  // couleur d'arrière plan
  )
```

Paramètres

value

[in] Couleur d'arrière plan.

© 2000-2024, MetaQuotes Ltd.

ColorBorder (méthode Get)

Retourne la couleur des bordures.

```
uint  ColorBorder()
```

Valeur de Retour

Couleur des bordures.

ColorBorder (méthode Set)

Définit la couleur des bordures.

```
void  ColorBorder(
  const uint  value,  // couleur des bordures
  )
```

Paramètres

value

[in] Couleur des bordures.

© 2000-2024, MetaQuotes Ltd.

ColorText (méthode Get)

Retourne la couleur du texte.

```
uint  ColorText()
```

Valeur de Retour

Couleur du texte.

ColorText (méthode Set)

Définit la couleur du texte.

```
void  ColorText(
  const uint  value,  // couleur du texte
  )
```

Paramètres

value

[in] Couleur du texte.

© 2000-2024, MetaQuotes Ltd.

ColorGrid (méthode Get)

Retourne la couleur de la grille.

```
uint  ColorGrid()
```

Valeur de Retour

Couleur de la grille.

ColorGrid (méthode Set)

Définit la couleur de la grille.

```
void  ColorGrid(
  const uint  value,  // couleur de la grille
  )
```

Paramètres

value

[in] Couleur de la grille.

© 2000-2024, MetaQuotes Ltd.

MaxData (méthode Get)

Retourne la quantité maximum de données (séries) autorisées.

```
uint  MaxData()
```

Valeur de Retour

La quantité maximum de données (séries).

MaxData (méthode Set)

Définit la quantité maximum de données (séries) autorisées.

```
void  MaxData(
  const uint  value,  // quantité de données
  )
```

Paramètres

value

[in] La quantité maximum de données (séries).

© 2000-2024, MetaQuotes Ltd.

MaxDescrLen (méthode Get)

Retourne la longueur maximum des descripteurs.

```
uint  MaxDescrLen()
```

Valeur de Retour

La valeur de la longueur maximum des descripteurs.

MaxDescrLen (méthode Set)

Définit la longueur maximum des descripteurs.

```
void  MaxDescrLen(
  const uint  value,  // longueur maximum
  )
```

Paramètres

value

[in] La valeur de la longueur maximum des descripteurs.

© 2000-2024, MetaQuotes Ltd.

ShowFlags (méthode Get)

Retourne la visibilité des élements du graphique.

```
bool ShowFlags()
```

Valeur de Retour

Visibilité des élements du graphique.

ShowFlags (méthode Set)

Définit la visibilité des élements du graphique.

```
void ShowFlags(
  const uint flags,  // flag
  )
```

Paramètres

flags

[in] Visibilité des éléments du graphique.

© 2000-2024, MetaQuotes Ltd.

IsShowLegend

Retourne et définit la visibilité de la légende sur le graphique.

```
bool IsShowLegend()
```

Valeur de Retour

true si la légende est visible, sinon — false.

© 2000-2024, MetaQuotes Ltd.

IsShowScaleLeft

Retourne la visibilité de l'échelle des valeurs sur la gauche.

```
bool IsShowScaleLeft()
```

Valeur de Retour

true si l'échelle des valeurs est visible, sinon — false.

© 2000-2024, MetaQuotes Ltd.

IsShowScaleRight

Retourne la visibilité de l'échelle des valeurs sur la droite.

```
bool  IsShowScaleRight()
```

Valeur de Retour

true si l'échelle des valeurs est visible, sinon — false.

© 2000-2024, MetaQuotes Ltd.

IsShowScaleTop

Retourne la visibilité de l'échelle des valeurs en haut.

```
bool IsShowScaleTop()
```

Valeur de Retour

true si l'échelle des valeurs est visible, sinon — false.

© 2000-2024, MetaQuotes Ltd.

IsShowScaleBottom

Retourne la visibilité de l'échelle des valeurs en bas.

```
bool  IsShowScaleBottom()
```

Valeur de Retour

true si l'échelle des valeurs est visible, sinon — false.

© 2000-2024, MetaQuotes Ltd.

IsShowGrid

Retourne la visibilité de la grille sur le graphique.

```
bool IsShowGrid()
```

Valeur de Retour

true si la grille est visible, sinon — false.

© 2000-2024, MetaQuotes Ltd.

IsShowDescriptors

Retourne la visibilité des descripteurs sur le graphique.

```
bool IsShowDescriptors()
```

Valeur de Retour

true si les descripteurs sont visibles, sinon — false.

© 2000-2024, MetaQuotes Ltd.

IsShowPercent

Retourne la visibilité des pourcentages sur le graphique.

```
bool IsShowPercent()
```

Valeur de Retour

true si les pourcentages sont visibles, sinon — false.

© 2000-2024, MetaQuotes Ltd.

VScaleMin (méthode Get)

Retourne la valeur minimum de l'échelle verticale de valeurs.

```
double  VScaleMin()
```

Valeur de Retour

La valeur minimum sur l'échelle verticale.

VScaleMin (méthode Set)

Définit la valeur minimum sur l'échelle verticale.

```
void  VScaleMin(
  const double  value,    // valeur sur l'échelle verticale
  )
```

Paramètres

value

[in] La valeur minimum.

© 2000-2024, MetaQuotes Ltd.

VScaleMax

Retourne la valeur maximum de l'échelle verticale de valeurs.

```
double  VScaleMax()
```

Valeur de Retour

La valeur maximum sur l'échelle verticale.

VScaleMax

Définit la valeur maximum sur l'échelle verticale.

```
void  VScaleMax(
  const double  value,  //   valeur sur l'échelle verticale
  )
```

Paramètres

value

[in] La valeur maximum.

© 2000-2024, MetaQuotes Ltd.

NumGrid

Retourne le nombre de divisions de l'échelle verticale lors de l'affichage de la grille sur le graphique.

```
uint  NumGrid()
```

Valeur de Retour

Le nombre de divisions.

NumGrid

Définit le nombre de divisions de l'échelle verticale lors de l'affichage de la grille sur le graphique..

```
void  NumGrid(
  const uint  value,  // nombre de divisions
  )
```

Paramètres

value

[in] Le nombre de divisions.

© 2000-2024, MetaQuotes Ltd.

DataOffset

Retourne la valeur de décalage des données.

```
int  DataOffset()
```

Valeur de Retour

Décalage des données.

DataOffset

Définit la valeur de décalage des données.

```
void  DataOffset(
  const int  value,  // décalage
  )
```

Paramètres

value

[in] Décalage des données.

© 2000-2024, MetaQuotes Ltd.

DataTotal

Retourne le nombre total de séries de données sur le graphique.

```
uint  DataTotal()
```

Valeur de Retour

Le nombre de séries.

© 2000-2024, MetaQuotes Ltd.

DrawDescriptors

Méthode virtuelle pour dessiner les descripteurs.

```
virtual void  DrawDescriptors()
```

© 2000-2024, MetaQuotes Ltd.

DrawData

Méthode virtuelle pour dessiner les séries de données à l'indice spécifié.

```
virtual void  DrawData(
   const uint  idx=0,  // index
   )
```

Paramètres

idx=0

[in] Indice des séries.

© 2000-2024, MetaQuotes Ltd.

Create

Méthode virtuelle pour créer une ressource graphique.

```
virtual bool  Create(
  const string       name,       // nom de la ressource
  const int          width,      // largeur
  const int          height,     // hauteur
  ENUM_COLOR_FORMAT  clrfmt,     // format
  )
```

Paramètres

name

[in] Base du nom d'une ressource graphique. Un nom de ressource est généré pendant la création en ajoutant une chaîne de caracctères pseudo-aléatoire.

width

[in] Largeur (taille suivant l'axe X) en pixels.

height

[in] Hauteur (taille suivant l'axe Y) en pixels.

clrfmt

[in] Méthode de traitement de la couleur. Voir la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de traitement des couleurs.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

AllowedShowFlags

Définit l'ensemble de flags de visibilité des élements du graphique.

```
void  AllowedShowFlags(
  const uint  flags,  // flags
  )
```

Paramètres

flags

[in] Flags autorisés.

© 2000-2024, MetaQuotes Ltd.

ShowLegend

Définit la visibilité de la légende (FLAG_SHOW_LEGEND).

```
void  ShowLegend(
  const bool  flag,  // valeur du flag
  )
```

Paramètres

flag

[in] Valeur du flag :

- true — la légende devient visible.
- false — la légende devient invisible.

© 2000-2024, MetaQuotes Ltd.

ShowScaleLeft

Définit la visibilité de l'échelle de gauche (FLAG_SHOW_SCALE_LEFT).

```
void  ShowScaleLeft(
  const bool  flag,  // valeur du flag
  )
```

Paramètres

flag

[in] Valeur du flag :

- true — l'échelle de gauche devient visible.
- false — l'échelle de gauche devient invisible.

© 2000-2024, MetaQuotes Ltd.

ShowScaleRight

Définit la visibilité de l'échelle de droite (FLAG_SHOW_SCALE_RIGHT).

```
void  ShowScaleRight(
  const bool  flag,  // valeur du flag
  )
```

Paramètres

flag

[in] Valeur du flag :

- true — l'échelle de droite devient visible.
- false — l'échelle de droite devient invisible.

© 2000-2024, MetaQuotes Ltd.

ShowScaleTop

Définit la visibilité de l'échelle du haut (FLAG_SHOW_SCALE_TOP).

```
void  ShowScaleTop(
  const bool  flag,  // valeur du flag
  )
```

Paramètres

flag

[in] Valeur du flag :

- true — l'échelle du haut devient visible.
- false — l'échelle du haut devient invisible.

© 2000-2024, MetaQuotes Ltd.

ShowScaleBottom

Définit la visibilité de l'échelle du bas (FLAG_SHOW_SCALE_BOTTOM).

```
void  ShowScaleBottom(
  const bool  flag,      // valeur du flag
  )
```

Paramètres

flag

[in] Valeur du flag :

- true — l'échelle du bas devient visible.
- false — l'échelle du bas devient invisible.

© 2000-2024, MetaQuotes Ltd.

ShowGrid

Définit la visibilité de la grille (FLAG_SHOW_GRID).

```
void  ShowGrid(
  const bool  flag,  // valeur du flag
  )
```

Paramètres

flag

[in] Valeur du flag :

- true — la grille devient visible.
- false — la grille devient invisible.

© 2000-2024, MetaQuotes Ltd.

ShowDescriptors

Définit la visibilité des descripteurs (FLAG_SHOW_DESCRIPTORS).

```
void  ShowDescriptors(
  const bool  flag,     // valeur du flag
  )
```

Paramètres

flag

[in] Valeur du flag :

- true — le descripteur devient visible.
- false — le descripteur devient invisible.

© 2000-2024, MetaQuotes Ltd.

ShowValue

Définit la visibilité des valeurs (FLAG_SHOW_VALUE).

```
void  ShowValue(
  const bool  flag,  // valeur du flag
  )
```

Paramètres

flag

[in] Valeur du flag :

- true — les valeurs deviennent visibles.
- false — les valeurs deviennent invisibles.

© 2000-2024, MetaQuotes Ltd.

ShowPercent

Définit la visibilité des pourcentages (FLAG_SHOW_PERCENT).

```
void  ShowPercent(
  const bool  flag,  // valeur du flag
  )
```

Paramètres

flag

[in] Valeur du flag :

- true — les pourcentages deviennent visibles.
- false — les pourcentages deviennent invisibles.

© 2000-2024, MetaQuotes Ltd.

LegendAlignment

Définit l'alignement du texte de la légende.

```
void  LegendAlignment(
  const ENUM_ALIGNMENT  value,  // flag
  )
```

Paramètres

value

[in] Prend l'une des valeurs de l'énumération ENUM_ALIGNMENT :

- ALIGNMENT_LEFT — alignement à gauche.
- ALIGNMENT_TOP — alignement en haut.
- ALIGNMENT_RIGHT — alignement à droite.
- ALIGNMENT_BOTTOM — alignement en bas.

© 2000-2024, MetaQuotes Ltd.

Accumulative

Définit le flag d'accumulation des valeurs pour les séries.

```
void  Accumulative(
  const bool  flag=true,  // valeur du flag
  )
```

Paramètres

flag=true

[in] Valeur du flag :

- true — la valeur actuelle des séries est remplacée par la somme de toutes les valeurs précédentes.
- false — le mode standard de dessin des séries.

© 2000-2024, MetaQuotes Ltd.

VScaleParams

Définit les paramètres de l'échelle verticale des valeurs.

```
void  VScaleParams(
  const double  max,    // maximum
  const double  min,    // minimum
  const uint    grid,   // nombre de divisions
  )
```

Paramètres

max

[in] La valeur minimum.

min

[in] La valeur maximum.

grid

[in] Le nombre de divisions d'échelle.

© 2000-2024, MetaQuotes Ltd.

DescriptorUpdate

Met à jour la valeur du descripteur de séries (à la position spécifiée).

```
bool  DescriptorUpdate(
  const uint    pos,    // index
  const string  descr,  // valeur
  )
```

Paramètres

pos

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

descr

[in] Valeur du descripteur.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

ColorUpdate

Met à jour les couleurs des séries (à la position spécifiée).

```
bool  ColorUpdate(
  const uint  pos,  // index
  const uint  clr,  // couleur
  )
```

Paramètres

pos

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

clr

[in] Couleur.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

ValuesCheck

Méthode virtuelle supplémentaire, effectue des calculs internes pour le dessin du graphique.

```
virtual void  ValuesCheck()
```

© 2000-2024, MetaQuotes Ltd.

Redraw

Méthode virtuelle pour redessiner le graphique.

```
virtual void  Redraw()
```

© 2000-2024, MetaQuotes Ltd.

DrawBackground

Méthode virtuelle pour redessiner l'arrière plan.

```
virtual void  DrawBackground()
```

© 2000-2024, MetaQuotes Ltd.

DrawLegend

Méthode virtuelle pour redessiner la légende.

```
virtual void  DrawLegend()
```

© 2000-2024, MetaQuotes Ltd.

DrawLegendVertical

Dessine une légende verticale.

```
int  DrawLegendVertical(
  const int  w,  // largeur
  const int  h,  // hauteur
  )
```

Paramètres

w

[in] La largeur maximum du texte dans la légende.

h

[in] La hauteur maximum du texte dans la légende.

Valeur de Retour

Largeur de la légende en pixels.

© 2000-2024, MetaQuotes Ltd.

DrawLegendHorizontal

Dessine une légende horizontale.

```
int  DrawLegendHorizontal(
  const int  w,  //
  const int  h,  //
  )
```

Paramètres

w

[in] La largeur maximum du texte dans la légende.

h

[in] La hauteur maximum du texte dans la légende.

Valeur de Retour

Hauteur de la légende en pixels.

© 2000-2024, MetaQuotes Ltd.

CalcScales

Méthode virtuelle de calcul des coordonnées des étiquettes de l'échelle des valeurs.

```
virtual void  CalcScales()
```

© 2000-2024, MetaQuotes Ltd.

DrawScales

Méthode virtuelle pour redessiner toutes les échelles de valeurs.

```
virtual void  DrawScales()
```

© 2000-2024, MetaQuotes Ltd.

DrawScaleLeft

Méthode virtuelle pour redessiner l'échelle gauche de valeurs.

```
virtual int  DrawScaleLeft(
  const bool  draw,  // flag
  )
```

Paramètres

draw

[in] Flag indiquant si l'échelle doit être redessinée.

Valeur de Retour

Largeur de l'échelle de valeurs.

© 2000-2024, MetaQuotes Ltd.

DrawScaleRight

Méthode virtuelle pour redessiner l'échelle droite de valeurs.

```
virtual int  DrawScaleRight(
  const bool  draw,  // flag
  )
```

Paramètres

draw

[in] Flag indiquant si l'échelle doit être redessinée.

Valeur de Retour

Largeur de l'échelle de valeurs.

© 2000-2024, MetaQuotes Ltd.

DrawScaleTop

Méthode virtuelle pour redessiner l'échelle de valeurs du haut.

```
virtual int  DrawScaleTop(
  const bool  draw,  // flag
  )
```

Paramètres

draw

[in] Flag indiquant si l'échelle doit être redessinée.

Valeur de Retour

Hauteur des l'échelle de valeurs.

© 2000-2024, MetaQuotes Ltd.

DrawScaleBottom

Méthode virtuelle pour redessiner l'échelle de valeurs du bas.

```
virtual int  DrawScaleBottom(
  const bool  draw,  // flag
  )
```

Paramètres

draw

[in] Flag indiquant si l'échelle doit être redessinée.

Valeur de Retour

Hauteur des l'échelle de valeurs.

© 2000-2024, MetaQuotes Ltd.

DrawGrid

Méthode virtuelle pour redessiner la grille.

```
virtual void  DrawGrid()
```

© 2000-2024, MetaQuotes Ltd.

DrawChart

Méthode virtuelle pour redessiner le graphique.

```
virtual void  DrawChart()
```

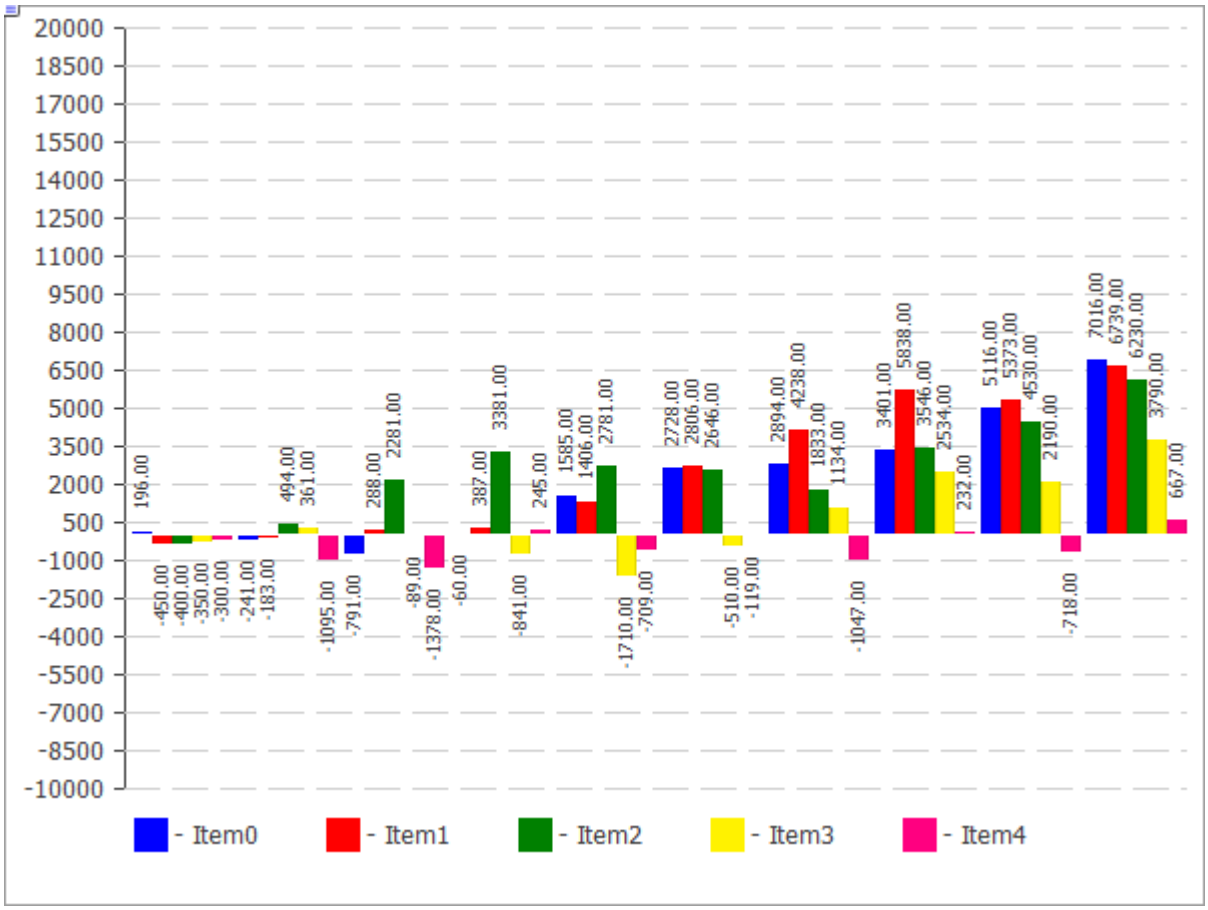
© 2000-2024, MetaQuotes Ltd.

CHistogramChart

Classe pour le dessin d'histogrammes.

Description

Toutes les méthodes permettant de dessiner des histogrammes sont implémentées dans cette classe. Elles peuvent être utilisées pour définir la largeur des barres et pour configurer l'utilisation des séries de données. Les méthodes de remplissage en dégradé des barres de l'histogramme sont incluses, permettant de visualiser les données plus clairement.



Le code de la figure ci-dessus est fourni ci-dessous.

Déclaration

```
class CHistogramChart : public CChartCanvas
```

Titre

```
#include <Canvas\Charts\HistogramChart.mqh>
```

Hiérarchie d'héritage

CCanvas
- CChartCanvas
  - CHistogramChart

© 2000-2024, MetaQuotes Ltd.

Méthodes de la classe

Méthode	Action
Gradient	Définit l'utilisation ou non du remplissage en dégradé des barres de l'histogramme.
BarGap	Définit la valeur de décalage de l'histogramme depuis l'origine.
BarMinSize	Définit la largeur minimum des barres de l'histogramme.
BarBorder	Définit le flag indiquant si les bordures de barres doivent être dessinées ou pas.
Create	Méthode virtuelle pour créer une ressource graphique.
SeriesAdd	Ajoute une nouvelle série de données.
SeriesInsert	Insère des série de données sur le graphique.
SeriesUpdate	Met à jour des séries de données sur le graphique.
SeriesDelete	Supprime des séries de données du graphique.
ValueUpdate	Met à jour la valeur d'un élément des séries spécifiées.
DrawData	Méthode virtuelle permettant de dessiner un histogramme pour les séries spécifiées.
DrawBar	Dessine une barre d'histogramme sous la forme d'un rectangle plein.
GradientBrush	Crée une brosse pour le remplissage en dégradé.

Méthodes héritées de la classe CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Méthodes héritées de la classe CChartCanvas

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

Exemple

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          HistogramChartSample.mq5 |
//|                    Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                                http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Exemple d'utilisation d'un histogramme"
//---
#include <Canvas\Charts\HistogramChart.mqh>
//+------------------------------------------------------------------+
//| entrées                                                          |
//+------------------------------------------------------------------+
input bool Accumulative=true;
//+------------------------------------------------------------------+
//| Fonction de démarrage du script                                  |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- crée le graphique
   CHistogramChart chart;
   if(!chart.CreateBitmapLabel("SampleHistogramChart",10,10,600,450))
     {
      Print("Erreur de création de l'histogramme : ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,20);
   chart.ShowValue(true);
   chart.ShowScaleTop(false);
   chart.ShowScaleBottom(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- joue avec les valeurs
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

```
//--- fin
   chart.Destroy();
   return(0);
  }
```

© 2000-2024, MetaQuotes Ltd.

Gradient

Définit l'utilisation ou non du remplissage en dégradé des barres de l'histogramme.

```
void  Gradient(
  const bool  flag=true,  // valeur du flag
  )
```

Paramètres

flag=true

Valeur du flag : true si le remplissage en dégradé est activé, sinon — false.

© 2000-2024, MetaQuotes Ltd.

BarGap

Définit la valeur de décalage de l'histogramme depuis l'origine.

```
void  BarGap(
  const uint  value,  // décalage
  )
```

Paramètres

value

[in] Valeur du décalage de l'histogramme.

© 2000-2024, MetaQuotes Ltd.

BarMinSize

Définit la largeur minimum des barres de l'histogramme.

```
void  BarMinSize(
  const uint  value,  // largeur minimum
  )
```

Paramètres

value

[in] la largeur minimum.

© 2000-2024, MetaQuotes Ltd.

BarBorder

Définit le flag indiquant si les bordures de barres doivent être dessinées ou pas.

```
void  BarBorder(
  const uint  value,  // flag
  )
```

Paramètres

value

[in] Valeur du flag :

- true — les bordures seront dessinées
- false — les bordures ne seront pas dessinées

© 2000-2024, MetaQuotes Ltd.

Create

Méthode virtuelle pour créer une ressource graphique.

```
virtual bool  Create(
  const string       name,     // nom
  const int          width,    // largeur
  const int          height,   // hauteur
  ENUM_COLOR_FORMAT  clrfmt,   // format
  )
```

Paramètres

name

[in] Base du nom d'une ressource graphique. Un nom de ressource est généré pendant la création en ajoutant une chaîne de caracctères pseudo-aléatoire.

width

[in] Largeur (taille suivant l'axe X) en pixels.

height

[in] Hauteur (taille suivant l'axe Y) en pixels.

clrfmt

[in] Méthode de traitement de la couleur. Voir la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de traitement des couleurs.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

SeriesAdd

Ajoute une nouvelle série de données.

```
bool  SeriesAdd(
  const double&  value[],  // valeurs
  const string  descr,     // label
  const uint    clr,       // couleur
  )
```

Paramètres

value[]

[in] Séries de données.

descr

[in] Etiquettes des séries.

clr

[in] Couleur d'affichage des séries.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

SeriesInsert

Insère des série de données sur le graphique.

```
bool  SeriesInsert(
  const uint    pos,        // index
  const double&  value[],   // valeurs
  const string  descr,      // label
  const uint    clr,        // couleur
  )
```

Paramètres

pos

[in] Indice pour l'insertion.

value[]

[in] Séries de données.

descr

[in] Etiquettes des séries.

clr

[in] Couleur d'affichage des séries.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

SeriesUpdate

Met à jour les séries de données sur le graphique.

```
bool  SeriesUpdate(
  const uint    pos,        // index
  const double  &value[],   // valeurs
  const string  descr,      // label
  const uint    clr,        // couleur
  )
```

Paramètres

pos

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

&value[]

[in] Nouvelles valeurs pour les séries de données.

descr

[in] Etiquettes des séries.

clr

[in] Couleur d'affichage des séries.

Valeur de Retour

true en cas de succès, sinon false.

© 2000-2024, MetaQuotes Ltd.

SeriesDelete

Supprime des séries de données du graphique.

```
bool  SeriesDelete(
  const uint  pos,  // index
  )
```

Paramètres

pos

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

ValueUpdate

Met à jour la valeur spécifiée dans la série spécifiée.

```
bool  ValueUpdate(
  const uint  series,  // indice de la série
  const uint  pos,     // indice de l'élément
  double      value,   // valeur
  )
```

Paramètres

series

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

pos

[in] Indice de l'élément dans la série.

value

[in] Nouvelle valeur.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

DrawData

Méthode virtuelle permettant de dessiner un histogramme pour les séries spécifiées.

```
virtual void  DrawData(
  const uint  index,  // index
  )
```

Paramètres

index

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

© 2000-2024, MetaQuotes Ltd.

DrawBar

Dessine une barre d'histogramme sous la forme d'un rectangle plein.

```
void  DrawBar(
  const int   x,    // coordonnée X
  const int   y,    // coordonnée Y
  const int   w,    // largeur
  const int   h,    // hauteur
  const uint  clr,  // couleur
  )
```

Paramètres

x

[in] Coordonnée X du point supérieur gauche du rectangle.

y

[in] Coordonnée Y du point supérieur gauche du rectangle.

w

[in] La largeur du rectangle.

h

[in] La hauteur du rectangle.

clr

[in] La couleur du rectangle.

© 2000-2024, MetaQuotes Ltd.

GradientBrush

Crée une brosse pour le remplissage en dégradé.

```
void  GradientBrush(
  const int   size,       // taille
  const uint  fill_clr,   // couleur de remplissage
  )
```

Paramètres

size

[in] Epaisseur de la brosse

fill_clr

[in] Couleur de remplissage

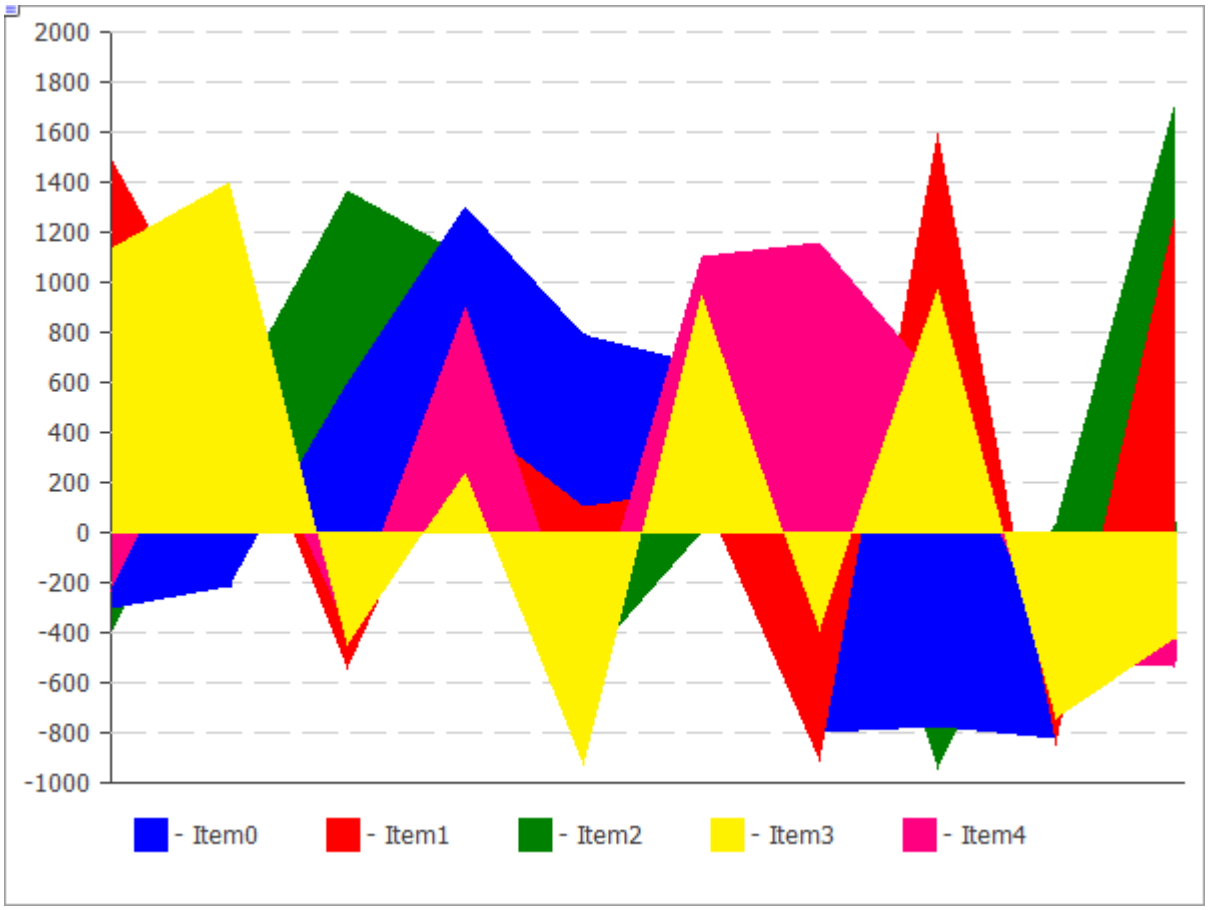
© 2000-2024, MetaQuotes Ltd.

CLineChart

Classe permettant de dessiner des courbes.

Description

Les méthodes de cette classe permettent de travailler avec les courbes sur le graphique. Elle comprennent la possibilité de remplir la zone délimitée par la courbe dessinée.



Le code de la figure ci-dessus est fourni ci-dessous.

Déclaration

```
class CLineChart : public CChartCanvas
```

Titre

```
#include <Canvas\Charts\LineChart.mqh>
```

Hiérarchie d'héritage

CCanvas
  CChartCanvas
    CLineChart

Méthodes de la classe

© 2000-2024, MetaQuotes Ltd.

Méthode	Action
Filled	Définit le flag de remplissage de la zone située sous la courbe définie par la série de données.
Create	Crée une ressource graphique.
SeriesAdd	Ajoute une nouvelle série de données.
SeriesInsert	Insère des série de données sur le graphique.
SeriesUpdate	Met à jour des séries de données sur le graphique.
SeriesDelete	Supprime des séries de données du graphique.
ValueUpdate	Met à jour la valeur spécifiée dans la série spécifiée.
DrawChart	Methode virtuelle permettant de dessiner une courbe et tous ses éléments.
DrawData	Méthode virtuelle permettant de dessiner une courbe pour les séries spécifiées.
CalcArea	Calcule la zone située sous la courbe définie par la série de données.

Méthodes héritées de la classe CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Méthodes héritées de la classe CChartCanvas

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

Exemple

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                LineChartSample.mq5 |
//|                    Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                                http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Exemple d'utilisation d'un graphique linéaire"
//---
#include <Canvas\Charts\LineChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=false;
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- crée le graphique
   CLineChart chart;
//--- crée le graphique
   if(!chart.CreateBitmapLabel("SampleHistogrammChart",10,10,600,450))
     {
      Print("Erreur de création du graphique linéaire : ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,15);
   chart.ShowScaleTop(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   chart.Filled();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- joue avec les valeurs
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- fin
   chart.Destroy();
   return(0);
  }
```

© 2000-2024, MetaQuotes Ltd.

Filled

Définit le flag indiquant s'il est nécessaire de remplir la zone située sous la courbe définie par la série de données.

```
void  Filled(
  const bool  flag=true,  // flag
  )
```

Paramètres

flag=true

[in] Valeur du flag :

- true — remplit la zone située sous la courbe
- false — ne remplit pas la zone située sous la courbe

© 2000-2024, MetaQuotes Ltd.

Create

Méthode virtuelle pour créer une ressource graphique.

```
virtual bool  Create(
  const string       name,     // nom
  const int          width,    // largeur
  const int          height,   // hauteur
  ENUM_COLOR_FORMAT  clrfmt,   // format
  )
```

Paramètres

name

[in] Base du nom d'une ressource graphique. Un nom de ressource est généré pendant la création en ajoutant une chaîne de caracctères pseudo-aléatoire.

width

[in] Largeur (taille suivant l'axe X) en pixels.

height

[in] Hauteur (taille suivant l'axe Y) en pixels.

clrfmt

[in] Méthode de traitement de la couleur. Voir la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de traitement des couleurs.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

SeriesAdd

Ajoute une nouvelle série de données.

```
bool  SeriesAdd(
  const double&  value[],  // valeurs
  const string  descr,     // label
  const uint    clr,       // couleur
  )
```

Paramètres

value[]

[in] Séries de données.

descr

[in] Etiquettes des séries.

clr

[in] Couleur d'affichage des séries.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

SeriesInsert

Insère des série de données sur le graphique.

```
bool  SeriesInsert(
  const uint     pos,        // index
  const double&  value[],    // valeurs
  const string   descr,      // label
  const uint     clr,        // color
  )
```

Paramètres

pos

[in] Indice pour l'insertion.

value[]

[in] Séries de données.

descr

[in] Etiquettes des séries.

clr

[in] Couleur d'affichage des séries.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

SeriesUpdate

Met à jour les séries de données sur le graphique.

```
bool  SeriesUpdate(
  const uint     pos,        // index
  const double&  value[],    // valeurs
  const string   descr,      // label
  const uint     clr,        // color
  )
```

Paramètres

pos

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

value[]

[in] Nouvelles valeurs pour les séries de données.

descr

[in] Etiquettes des séries.

clr

[in] Couleur d'affichage des séries.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

SeriesDelete

Supprime des séries de données du graphique.

```
bool  SeriesDelete(
  const uint  pos,  // index
  )
```

Paramètres

pos

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

ValueUpdate

Met à jour la valeur spécifiée dans la série spécifiée.

```
bool  ValueUpdate(
  const uint  series,  // indice de la série
  const uint  pos,     // indice de l'élément
  double      value,   // valeur
  )
```

Paramètres

series

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

pos

[in] Indice de l'élément dans la série.

value

[in] Nouvelle valeur.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

DrawChart

Methode virtuelle permettant de dessiner une courbe et tous ses éléments.

```
virtual void  DrawChart()
```

© 2000-2024, MetaQuotes Ltd.

DrawData

Méthode virtuelle permettant de dessiner une courbe pour les séries spécifiées.

```
virtual void  DrawData(
  const uint  index,  // index
  )
```

Paramètres

index

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

© 2000-2024, MetaQuotes Ltd.

CalcArea

Calcule la zone située sous la courbe définie par la série de données.

```
double  CalcArea(
  const uint  index,  // index
  )
```

Paramètres

index

[in] Indice des séries — le numéro de série pour son ajout, commençant à 0.

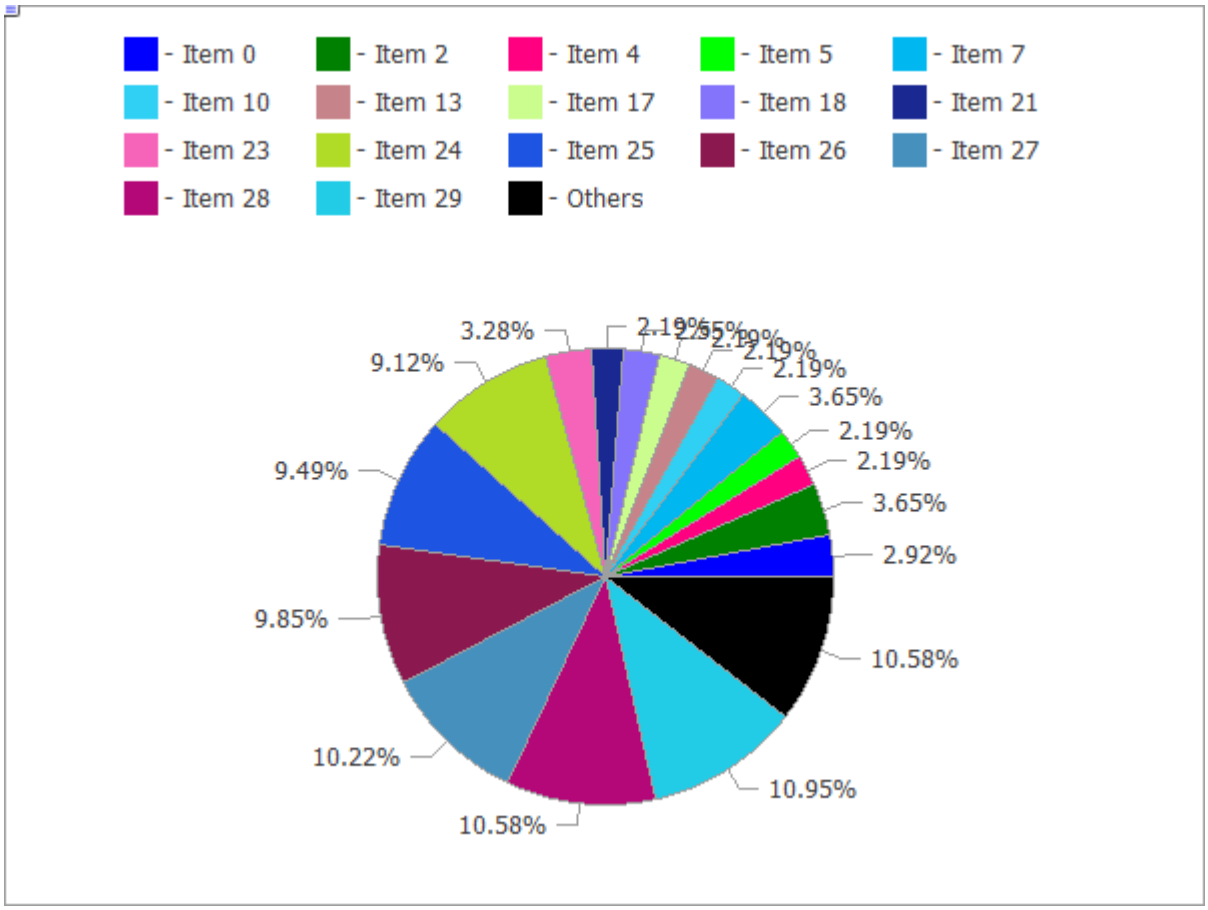
Valeur de Retour

Zone de la figure limitée par la courbe définie par la série de données.

© 2000-2024, MetaQuotes Ltd.

CPieChart

Classe pour dessiner des diagrammes circulaires.



Le code de la figure ci-dessus est fourni ci-dessous.

Description

Les méthodes de cette classe sont conçues pour une utilisation de A à Z des diagrammes circulaires, de la création d'une ressource graphique à la création des étiquettes pour les segments.

Déclaration

```
class CPieChart : public CChartCanvas
```

Titre

```
#include <Canvas\Charts\PieChart.mqh>
```

Hiérarchie d'héritage

CCanvas
  CChartCanvas
    CPieChart

Méthodes de la classe

© 2000-2024, MetaQuotes Ltd.

Méthode	Action
Create	Méthode virtuelle pour créer une ressource graphique.
SeriesSet	Définit une série de valeurs devant être affichées sur le diagramme circulaire.
ValueAdd	Ajoute une nouvelle valeur à un diagramme circulaire (ajout à la fin).
ValueInsert	Insère une nouvelle valeur à un diagramme circulaire (à la position spécifiée).
ValueUpdate	Met à jour la valeur d'un diagramme circulaire (à la position spécifiée).
ValueDelete	Supprime une valeur d'un diagramme circulaire (à la position spécifiée).
DrawChart	Méthode virtuelle dessinant un diagramme circulaire et tous ses éléments.
DrawPie	Dessine le segment d'un diagramme circulaire correspondant à une valeur spécifiée.
LabelMake	Génère l'étiquette d'un segment basée sur sa valeur et sur son étiquette d'origine.

Méthodes héritées de la classe CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Méthodes héritées de la classe CChartCanvas

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

© 2000-2024, MetaQuotes Ltd.

Exemple

```
//+------------------------------------------------------------------+
//|                                                PieChartSample.mq5 |
//|                   Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                             http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Exemple d'utilisation d'un diagramme circulaire"
//---
#include <Canvas\Charts\PieChart.mqh>
//+------------------------------------------------------------------+
//| entrйes                                                          |
//+------------------------------------------------------------------+
input int      Width=600;
input int      Height=450;
//+------------------------------------------------------------------+
//| Fonction de dйmarrage du script                                  |
//+------------------------------------------------------------------+
int OnStart(void)
  {
//--- vйrifications
   if(Width<=0 || Height<=0)
     {
     Print("Trop simple.");
      return(-1);
     }
//--- crйe le graphique
   CPieChart pie_chart;
   if(!pie_chart.CreateBitmapLabel("PieChart",10,10,Width,Height))
     {
     Print("Erreur de crйation du diagramme circulaire : ",GetLastError());
      return(-1);
     }
   pie_chart.ShowPercent();
//--- dessin
   for(uint i=0;i<30;i++)
     {
      pie_chart.ValueAdd(100*(i+1),"Item "+IntegerToString(i));
      Sleep(10);
     }
   Sleep(2000);
//--- dйsactive la lйgende
   pie_chart.LegendAlignment(ALIGNMENT_LEFT);
   Sleep(2000);
//--- dйsactive la lйgende
   pie_chart.LegendAlignment(ALIGNMENT_RIGHT);
   Sleep(2000);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- dйsactive la lйgende
   pie_chart.LegendAlignment(ALIGNMENT_TOP);
   Sleep(2000);
//--- dйsactive la lйgende
   pie_chart.ShowLegend(false);
   Sleep(2000);
//--- dйsactive les pourcentages
   pie_chart.ShowPercent(false);
   Sleep(2000);
//--- dйsactive lees descripteurs
   pie_chart.ShowDescriptors(false);
   Sleep(2000);
//--- active tout
   pie_chart.ShowLegend();
   pie_chart.ShowValue();
   pie_chart.ShowDescriptors();
   Sleep(2000);
//--- ou de cette faзon
   pie_chart.ShowFlags(FLAG_SHOW_LEGEND|FLAG_SHOW_DESCRIPTORS|FLAG_SHOW_PERCENT);
   uint total=pie_chart.DataTotal();
//--- joue avec les valeurs
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ValueUpdate(i,100*(rand()%10+1));
      Sleep(1000);
     }
//--- joue avec les couleurs
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ColorUpdate(i%total,RandomRGB());
      Sleep(1000);
     }
//--- rotation
   while(!IsStopped())
     {
      pie_chart.DataOffset(pie_chart.DataOffset()+1);
      Sleep(200);
     }
//--- fin
   pie_chart.Destroy();
   return(0);
  }
//+------------------------------------------------------------------+
//| Couleur RGB alйatoire                                            |
//+------------------------------------------------------------------+
uint RandomRGB(void)
  {
   return(XRGB(rand()%255,rand()%255,rand()%255));
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Méthode virtuelle pour créer une ressource graphique.

```
virtual bool  Create(
  const string       name,     // nom
  const int          width,    // largeur
  const int          height,   // hauteur
  ENUM_COLOR_FORMAT  clrfmt,   // format
  )
```

Paramètres

name

[in] Base du nom d'une ressource graphique. Un nom de ressource est généré pendant la création en ajoutant une chaîne de caracctères pseudo-aléatoire.

width

[in] Largeur (taille suivant l'axe X) en pixels.

height

[in] Hauteur (taille suivant l'axe Y) en pixels.

clrfmt

[in] Méthode de traitement de la couleur. Voir la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de traitement des couleurs.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

SeriesSet

Définit une série de valeurs devant être affichées sur le diagramme circulaire.

```
bool  SeriesSet(
  const double&  value[],  // valeurs
  const string&  text[],   // labels
  const uint&    clr[],    // couleur
  )
```

Paramètres

value[]

[in] Tableau de valeurs.

text[]

[in] Tableau des textes des valeurs.

clr[]

[in] Tableau des couleurs des valeurs.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

ValueAdd

Ajoute une nouvelle valeur à un diagramme circulaire (ajout à la fin).

```
bool  ValueAdd(
  const double  value,  // valeur
  const string  descr,  // label
  const uint    clr,    // couleur
  )
```

Paramètres

value

[in] Valeur.

descr

[in] Texte de la valeur.

clr

[in] Couleur de la valeur.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

ValueInsert

Insère une nouvelle valeur à un diagramme circulaire (à la position spécifiée).

```
bool  ValueInsert(
  const uint    pos,    // index
  const double  value,  // valeur
  const string  descr,  // label
  const uint    clr,    // couleur
  )
```

Paramètres

pos

[in] Indice pour l'insertion.

value

[in] Valeur.

descr

[in] Texte de la valeur.

clr

[in] Couleur de la valeur.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

ValueUpdate

Met à jour la valeur d'un diagramme circulaire (à la position spécifiée).

```
bool  ValueUpdate(
  const uint    pos,    // index
  const double  value,  // valeur
  const string  descr,  // label
  const uint    clr,    // couleur
  )
```

Paramètres

pos

[in] Indice de la valeur — le numéro dans la série, commençant à 0.

value

[in] Valeur.

descr

[in] Texte de la valeur.

clr

[in] Couleur de la valeur.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

ValueDelete

Supprime une valeur d'un diagramme circulaire (à la position spécifiée).

```
bool  ValueDelete(
  const uint  pos,  // index
  )
```

Paramètres

pos

[in] Indice de la valeur — le numéro dans la série, commençant à 0.

Valeur de Retour

true en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

DrawChart

Méthode virtuelle dessinant un diagramme circulaire et tous ses éléments.

```
virtual void  DrawChart()
```

© 2000-2024, MetaQuotes Ltd.

DrawPie

Dessine le segment d'un diagramme circulaire correspondant à une valeur spécifiée.

```
void  DrawPie(
  double      fi3,   // angle du rayon depuis le centre du diagramme, qui définit la
  double      fi4,   // angle du rayon depuis le centre du diagramme, qui définit la
  int         idx,   // index
  CPoint&     p[],  //
  const uint  clr,   //
  )
```

Paramètres

fi3

[in] Angle en radians, qui défini la première limite de l'arc.

fi4

[in] Angle en radians, qui définit la deuxième limite de l'arc.

idx

[in] Indice de la valeur à laquelle le segment correspond.

p[]

[in] Tableau de points de référence (x, y) pour dessiner les segments.

clr

[in] Couleur du segment.

© 2000-2024, MetaQuotes Ltd.

LabelMake

Génère l'étiquette d'un segment basée sur sa valeur et sur son étiquette d'origine.

```
string  LabelMake(
  const string  text,      // label
  const double  value,     // valeur
  const bool    to_left,   // flag
  )
```

Paramètres

text

[in] Texte de l'étiquette.

value

[in] Valeur.

to_left

[in] Définit l'ordre d'affichage de l'étiquette :

- true — texte, puis valeur.
- false — valeur, puis texte.

Valeur de Retour

Etiquette du segment.

© 2000-2024, MetaQuotes Ltd.

Graphiques 3D

La section présente les classes pour le développement graphique en trois dimensions. Les classes sont basées sur les fonctions pour travailler avec DirectX. CCnavas3D est une classe de base contenant les méthodes de gestion de la caméra et de l'éclairage, ainsi que le gestionnaire de ressources graphiques - textures, shaders, buffers de sommets, indexes et paramètres d'ombrage.

Pour commencer à travailler avec la bibliothèque, lisez simplement l'article Comment créer des graphiques 3D à l'aide de DirectX dans MetaTrader 5.

En outre, il existe les classes des objets de base de la scène, comme une boîte, une surface tridimensionnelle sur les données utilisateur et une grille arbitraire.

© 2000-2024, MetaQuotes Ltd.

CCanvas3D

CCanvas3D est une classe permttant la création et la visualisation simplifiée d'objets 3D sur un graphique.

Description

CCanvas3D simplifie grandement la création et la visualisation de grandes quantités de données sous la forme de graphiques 3D animés. La classe contient des méthodes pour gérer la caméra et la lumière, ainsi qu'un gestionnaire de ressources pour créer des ressources graphiques : textures, shaders, vertex buffers, indexes, et paramètres d'ombrage.

En outre, la bibliothèque contient les classes des objets de base de la scène, comme une boîte, une surface tridimensionnelle sur les données utilisateur ou une grille arbitraire.

Votre carte vidéo doit supporter DX 11 et Shader Model 5.0 pour que les fonctions puissent fonctionner.

Déclaration

```
class CCanvas
```

Titre

```
#include <Canvas\Canvas.mqh>
```

Hiérarchie d'héritage

CCanvas
  CCanvas3D

Méthodes de classe par groupes

Création/suppression	Description
Create	Crée une ressource graphique pour effectuer le rendu d'une scène 3D sans être liée à un objet graphique.
Attach	Retourne une ressource graphique de l'objet OBJ_BITMAP_LABEL et l'attache à une instance de la classe CCanvas.
ObjectAdd	Ajoute un objet à une scène 3D pour pouvoir effectuer ensuite le rendu.
Destroy	Détruit une ressource graphique et relâche le contexte graphique 3D.
Lumière	
AmbientColorSet	Définit la couleur et l'intensité de la lumière ambiante.
AmbientColorGet	Retourne la couleur et l'intensité de la lumière ambiante.

© 2000-2024, MetaQuotes Ltd.

Création/suppression	Description
LightDirectionSet	Définit la direction d'une source dirigée de lumière.
LightDirectionGet	Retourne la direction d'une source dirigée de lumière.
LightColorSet	Définit la couleur et l'intensité d'une source dirigée de lumière.
LightColorGet	Retourne la couleur et l'intensité d'une source dirigée de lumière.
Caméra	
ProjectionMatrixSet	Calcule et définit la matrice de projection des coordonnées 3D vers un cadre 2D.
ProjectionMatrixGet	Retourne la matrice de projection d'une scène 3D vers un cadre 2D.
ViewMatrixSet	Définit la matrice de vue d'une scène 3D.
ViewMatrixGet	Retourne la matrice de vue d'une scène 3D.
ViewPositionSet	Définit le point de vue d'une scène 3D.
ViewRotationSet	Définit la direction du regard vers une scène 3D.
ViewTargetSet	Définit les coordonnées du point vers lequel le regard est porté.
ViewUpDirectionSet	Définit la direction de la bordure du cadre supérieur dans un espace 3D.
Rendu	
Render	Effectue le rendu de tous les objets de la scène du buffer interne du cadre pour pouvoir être affiché.
RenderBegin	Prépare un contexte graphique pour effectuer le rendu d'un nouveau cadre.
RenderEnd	Copie un cadre dont le rendu a été réalisé vers le buffer interne et met à jour l'image graphique si nécessaire.
Obtenir des ressources	
DXContext	Retourne le handle du contexte graphique.
DXDispatcher	Retourne le handle du gestionnaire de ressources.
InputScene	Retourne un pointeur sur le buffer des paramètres de la scène.

© 2000-2024, MetaQuotes Ltd.

AmbientColorGet

Retourne la couleur et l'intensité de la lumière ambiante.

```
void  AmbientColorGet(
   DXColor  &ambient_color      // couleur et intensité de la lumière d'ambiance
   );
```

Paramètres

&ambient_color

[out] Couleur de la lumière d'ambiance.

Valeur de Retour

Aucune

Note

L'intensité est stockée dans le canal alpha de la structure DXColor.

© 2000-2024, MetaQuotes Ltd.

AmbientColorSet

Définit la couleur et l'intensité de la lumière ambiante.

```
void  AmbientColorSet(
   const DXColor  &ambient_color      // couleur et intensité de la lumière d'ambiance
   );
```

Paramètres

&ambient_color

[in] Couleur de la lumière d'ambiance.

Valeur de Retour

Aucune

Note

L'intensité est définie dans le canal alpha de la structure DXColor.

© 2000-2024, MetaQuotes Ltd.

Attach

Retourne la ressource graphique de l'objet OBJ_BITMAP_LABEL et l'attache à une instance de la classe CCanvas.

```
bool  Attach(
   const long         chart_id,                          // identifiant du graphic
   const string       objname,                           // nom de l'objet
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // méthode de gestion des
   )
```

Crée une ressource graphique de l'objet OBJ_BITMAP_LABEL et l'attache à une instance de la classe CCanvas.

```
bool  Attach(
   const long         chart_id,                          // identifiant du graphic
   const string       objname,                           // nom de l'objet
   const int          width,                             // largeur de l'image en
   const int          height,                            // hauteur de l'image en
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // méthode de gestion des
   )
```

Paramètres

chart_id

[in] Identifiant du graphique.

objname

[in] Nom de l'objet graphique.

width

[in] Largeur du cadre dans une ressource.

height

[in] Hauteur du cadre.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Méthode de gestion des couleurs. Consultez la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de gestion des couleurs.

Note

true - en cas de succès, false si en cas d'échec de l'ajout de l'objet graphique.

© 2000-2024, MetaQuotes Ltd.

Create

Crée une ressource graphique pour effectuer le rendu d'une scène 3D sans être liée à un objet graphique.

```
virtual bool  Create(
   const string       name,                                // nom de l'objet graphiqu
   const int          width,                               // largeur
   const int          height,                              // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA     // format de la couleur
   );
```

Paramètres

name

[in] Nom de l'objet graphique.

width

[in] Largeur du cadre.

height

[in] Hauteur du cadre.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Méthode de gestion des couleurs. Consultez la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de gestion des couleurs.

Note

true - si la ressource est créée, false sinon.

© 2000-2024, MetaQuotes Ltd.

Destroy

Détruit une ressource graphique et relâche le contexte graphique 3D.

```
virtual void  Destroy()
```

Valeur de Retour

Aucune

Note

Si une ressource graphique était liée à un objet graphique, celui-ci sera supprimé aussi.

© 2000-2024, MetaQuotes Ltd.

DXContext

Retourne le handle du contexte graphique.

```
int  DXContext()
```

Valeur de Retour

Handle du contexte graphique.

© 2000-2024, MetaQuotes Ltd.

DXDispatcher

Retourne le handle du gestionnaire de ressources.

```
CDXDispatcher*  DXDispatcher()
```

Valeur de Retour

Handle du gestionnaire de ressources.

© 2000-2024, MetaQuotes Ltd.

InputScene

Retourne un pointeur sur le buffer des paramètres de la scène.

```
CDXInput*  InputScene()
```

Valeur de Retour

Pointeur vers le buffer des paramètres de la scène.

© 2000-2024, MetaQuotes Ltd.

LightColorGet

Retourne la couleur et l'intensité d'une source dirigée de lumière.

```
void  LightColorGet(
   DXColor  &light_color      // couleur et intensité de la lumière directionnelle
   );
```

Paramètres

&light_color

[out] Couleur et intensité de la lumière directionnelle.

Valeur de Retour

Aucune.

Note

L'intensité est stockée dans le canal alpha de la structure DXColor.

© 2000-2024, MetaQuotes Ltd.

LightColorSet

Définit la couleur et l'intensité d'une source dirigée de lumière.

```
void  LightColorSet(
   const DXColor  &light_color      // couleur et intensité de la lumière directionnel
   );
```

Paramètres

&light_color

[in] Couleur et intensité de la lumière directionnelle.

Valeur de Retour

Aucune.

Note

L'intensité est définie dans le canal alpha de la structure DXColor.

© 2000-2024, MetaQuotes Ltd.

LightDirectionGet

Retourne la direction d'une source dirigée de lumière.

```
void  LightDirectionGet(
   DXVector3  &light_direction      // vecteur de direction
   );
```

Paramètres

&light_direction

[out] Vecteur de direction.

Valeur de Retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

LightDirectionSet

Définit la direction d'une source dirigée de lumière.

```
void  LightDirectionSet(
   const DXVector3  &light_direction      // vecteur de direction
   );
```

Paramètres

&light_direction

[in] Vecteur de direction.

Valeur de Retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

ObjectAdd

Ajoute un objet à une scène 3D pour pouvoir effectuer ensuite le rendu.

```
bool  ObjectAdd(
   CDXObject  *object      // pointeur vers l'objet
   );
```

Paramètres

object

[in] Pointeur sur une instance de la classe dérivée de la classe abstraite CDXObject.

Valeur de Retour

true – en cas de succès, false - en cas d'échec d'ajout d'un objet graphique 3D.

© 2000-2024, MetaQuotes Ltd.

ProjectionMatrixGet

Retourne la matrice de projection d'une scène 3D vers un cadre 2D.

```
void  ProjectionMatrixGet(
   DXMatrix  &projection_matrix      // matrice de projection
   );
```

Paramètres

&projection_matrix

[out] Matrice de projection.

Valeur de Retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

ProjectionMatrixSet

Calcule et définit la matrice de projection des coordonnées 3D vers un cadre 2D.

```
void  ProjectionMatrixSet(
   float  fov,              // champ de vue
   float  aspect_ratio,     // rapport de l'aspect du cadre
   float  z_near,           //
   float  z_far             //
   );
```

Paramètres

fov

[in] Largeur du champ de vue en radians pour créer une projection de la scène.

aspect_ratio

[in] Rapport de l'aspect du cadre 2D.

z_near

[in] Distance par rapport au plan de coupe proche.

z_far

[in] Distance par rapport au plan de coupe lointain.

Valeur de Retour

Aucune.

Note

Le cadre 2D n'affiche que des projections des objets 3D tombant dans le champ spécifié de vue et localisé entre les plans de coupe proche et lointain.

© 2000-2024, MetaQuotes Ltd.

Render

Effectue le rendu de tous les objets de la scène du buffer interne du cadre pour pouvoir être affiché.

```
bool  Render(
   uint  flags,                    // combinaison de flags
   uint  background_color=0        //couleur d'arrière plan
   );
```

Paramètres

flags

[in] Combinaison de flags définissant le mode de rendu. Valeurs possibles :
DX_CLEAR_COLOR - efface le buffer d'image en utilisant *background_color*.
DX_CLEAR_DEPTH - efface le buffer de profondeur.

background_color=0

[in] Couleur d'arrière plan de la scène 3D.

Valeur de Retour

true - en cas de succès, false - en cas d'échec du rendu.

Note

L'appel à Render() ne met pas à jour la scène sur le graphique. Il ne fait que mettre à jour le buffer interne de l'image. La méthode Update() doit être appelée explicitement pour effectuer le rendu du cadre mis à jour.

Render() comprend les appels à RenderBegin et RenderEnd().

© 2000-2024, MetaQuotes Ltd.

RenderBegin

Prépare un contexte graphique pour effectuer le rendu d'un nouveau cadre.

```
virtual bool  RenderBegin(
   uint  flags,                  // combinaison de flags
   uint  background_color=0      //couleur d'arrière plan
   );
```

Paramètres

flags

[in] Combinaison de flags définissant le mode de rendu. Valeurs possibles :
DX_CLEAR_COLOR - efface le buffer d'image en utilisant *background_color*.
DX_CLEAR_DEPTH - efface le buffer de profondeur.

background_color=0

[in] Couleur d'arrière plan de la scène 3D.

Valeur de Retour

true - en cas de succès, false - en cas d'échec de la mise à jour des entrées du shader.

© 2000-2024, MetaQuotes Ltd.

RenderEnd

Copie un cadre dont le rendu a été réalisé vers le buffer interne et met à jour l'image graphique si nécessaire.

```
virtual bool  RenderEnd(
   bool  redraw=false      // flag de mise à jour
   );
```

Paramètres

redraw=false

[in] Flag spécifiant le besoin de redessiner un graphique.

Valeur de Retour

true - en cas de succès, false sinon.

© 2000-2024, MetaQuotes Ltd.

ViewMatrixGet

Retourne la matrice de vue d'une scène 3D.

```
void  ViewMatrixGet(
   DXMatrix  &view_matrix      // matrice de vue
   );
```

Paramètres

&view_matrix

[out] Matrice de vue définissant la position et la direction de la caméra dans l'espace 3D.

Valeur de Retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

ViewMatrixSet

Définit la matrice de vue d'une scène 3D.

```
void  ViewMatrixSet(
   const DXMatrix  &view_matrix      // matrice de vue
   );
```

Paramètres

&view_matrix

[in] Matrice de vue définissant la position et la direction de la caméra dans l'espace 3D.

Valeur de Retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

ViewPositionSet

Définit le point de vue d'une scène 3D.

```
void  ViewPositionSet(
   const DXVector3  &position      // position du point de vue
   );
```

Paramètres

&position

[in] Définition de la position du point de vue dans une scène 3D.

Valeur de Retour

Aucune.

Note

Changer la position d'un point de vue avec ViewPositionSet() change la matrice de vue obtenue avec ViewMatrixGet().

© 2000-2024, MetaQuotes Ltd.

ViewRotationSet

Définit la direction du regard vers une scène 3D.

```
void  ViewRotationSet(
   const DXVector3  &rotation      // vecteur d'angles de rotation
   );
```

Paramètres

&rotation

[in] Vecteur définissant les angles d'Euler pour calculer la direction du regard vers une scène 3D.

Valeur de Retour

Aucune.

Note

Définir la direction du regard avec ViewRotationSet() change la matrice de vue obtenue avec ViewMatrixGet().

© 2000-2024, MetaQuotes Ltd.

ViewTargetSet

Définit les coordonnées du point vers lequel le regard est porté.

```
void  ViewTargetSet(
   const DXVector3  &target      // coordonnées cible
   );
```

Paramètres

&target

[in] Coordonnées du point vers lequel le regard est porté.

Valeur de Retour

Aucune.

Note

Utilisé pour fixer le regard sur un point de la scène lors du déplacement du point de vue.

Définie une nouvelle coordonnée cible avec ViewRotationSet() change la matrice de vue obtenue avec ViewMatrixGet().

ViewTargetSet() est utilisée avec ViewUpDirectionSet() pour définir la direction du regard.

© 2000-2024, MetaQuotes Ltd.

ViewUpDirectionSet

Définit la direction de la bordure du cadre supérieur dans un espace 3D.

```
void  ViewUpDirectionSet(
   const DXVector3  &up_direction      // direction vers le haut
   );
```

Paramètres

&up_direction

[in] Direction de la partie supérieure du cadre dans l'espace 3D.

Valeur de Retour

Aucune.

Note

Définir une nouvelle direction avec ViewUpDirectionSet() change la matrice de vue obtenue avec ViewMatrixGet().

ViewUpDirectionSet() est utilisée avec ViewTargetSet() pour définir la direction du regard.

© 2000-2024, MetaQuotes Ltd.

CChart

La classe CChart est une classe permettant un accès simplifié aux propriétés de l'objet graphique "Chart".

Description

La classe CChart fournit un accès aux propriétés de l'objet "Chart".

Déclaration

```
class CChart : public CObject
```

Titre

```
#include <Charts\Chart.mqh>
```

Hiérarchie d'héritage

CObject
  CChart

Méthodes de Classe

Accès aux données protégées	
ChartID	Retourne l'identifiant du graphique
Propriétés générales	
Mode	Définit/Retourne la valeur de la propriété "Mode" (barre, bougie ou ligne)
Foreground	Définit/Retourne la valeur de la propriété "Foreground"
Shift	Définit/Retourne la valeur de la propriété "Shift"
ShiftSize	Définit/Retourne la valeur de la propriété "ShiftSize" (en pourcentage)
AutoScroll	Définit/Retourne la valeur de la propriété "AutoScroll"
Scale	Définit/Retourne la valeur de la propriété "Scale"
ScaleFix	Définit/Retourne la valeur de la propriété "ScaleFix" (échelle de graphique fixe ou pas)
ScaleFix_11	Définit/Retourne la valeur de la propriété "ScaleFix_11" (échelle de graphique 1:1, ou pas)
FixedMax	Définit/Retourne la valeur de la propriété "FixedMax" (prix maximum fixé)
FixedMin	Définit/Retourne la valeur de la propriété "FixedMin" (prix minimal fixé)

© 2000-2024, MetaQuotes Ltd.

Accès aux données protégées	
ScalePPB	Définit/Retourne la valeur de la propriété "ScalePPB" (l'échelle est "point par barre" ou pas)
PointsPerBar	Définit/Retourne la valeur de la propriété "PointsPerBar" (en points par barre)
Afficher les propriétés	
ShowOHLC	Définit/Retourne la valeur de la propriété "ShowOHLC"
ShowLineBid	Définit/Retourne la valeur de la propriété "ShowLineBid"
ShowLineAsk	Définit/Retourne la valeur de la propriété "ShowLineAsk"
ShowLastLine	Définit/Retourne la valeur de la propriété "ShowLastLine"
ShowPeriodSep	Définit/Retourne la valeur de la propriété "ShowPeriodSep" (afficher les séparateurs de période)
ShowGrid	Définit/Retourne la valeur de la propriété "ShowGrid"
ShowVolumes	Définit/Retourne la valeur de la propriété "ShowVolumes" (couleur des volumes et des niveaux des positions ouvertes)
ShowObjectDescr	Définit/Retourne la valeur de la propriété "ShowObjectDescr" (afficher la description des objets graphiques)
ShowDateScale	Définit/Retourne la valeur de la propriété "ShowDateScale" (échelle de date du graphique)
ShowPriceScale	Définit/Retourne la valeur de la propriété "ShowPriceScale" (échelle des prix du graphique)
Couleurs	
ColorBackground	Définit/Retourne la valeur de la propriété "ColorBackground" (couleur d'arrière plan du graphique)
ColorForeground	Définit/Retourne la valeur de la propriété "ColorForeground" (couleur des axes, de l'échelle et des textes OHLC)
ColorGrid	Définit/Retourne la valeur de la propriété "ColorGrid" (couleur de la grille)
ColorBarUp	Définit/Retourne la valeur de la propriété "ColorBarUp" (couleur des barres haussières, leurs ombres et le contour des corps des bougies)
ColorBarDown	Définit/Retourne la valeur de la propriété "ColorBarDown" (couleur des barres baissières, leurs ombres et le contour des corps des bougies)

© 2000-2024, MetaQuotes Ltd.

Accès aux données protégées	
ColorCandleBull	Définit/Retourne la valeur de la propriété "ColorCandleBull" (couleur du corps des bougies haussières)
ColorCandleBear	Définit/Retourne la valeur de la propriété "ColorCandleBear" (couleur du corps des bougies baissières)
ColorChartLine	Définit/Retourne la valeur de la propriété "ColorChartLine" (couleur des graphiques en ligne et des bougies en Doji)
ColorVolumes	Définit/Retourne la valeur de la propriété "ColorVolumes" (couleur des volumes et des niveaux des positions ouvertes)
ColorLineBid	Définit/Retourne la valeur de la propriété "ColorLineBid" (couleur de la ligne Bid)
ColorLineAsk	Définit/Retourne la valeur de la propriété "ColorLineAsk" (couleur de la ligne Ask)
ColorLineLast	Définit/Retourne la valeur de la propriété "ColorLineLast" (couleur de la ligne du dernier deal)
ColorStopLevels	Définit/Retourne la valeur de la propriété "ColorStopLevels" (couleur des niveaux de SL et TP)
Propriétés en lecture seule	
VisibleBars	Retourne le nombre total de barres visibles sur le graphique
WindowsTotal	Retourne le nombre total de fenêtres de graphiques, incluant les sous-fenêtres des indicateurs
WindowIsVisible	Retourne le flag de visibilité de la sous-fenêtre spécifiée du graphique
WindowHandle	Retourne le handler de la fenêtre du graphique (HWND)
FirstVisibleBar	Retourne le numéro de la première barre visible du graphique
WidthInBars	Retourne la largeur de la fenêtre en nombre de barres.
WidthInPixels	Retourne la largeur de la sous-fenêtre en pixels.
HeightInPixels	Retourne la hauteur de la sous-fenêtre en pixels.
PriceMin	Retourne le prix minimum de la sous-fenêtre spécifiée
PriceMax	Retourne le prix maximum de la sous-fenêtre spécifiée
Propriétés	
Attach	Assigne le graphique courant à l'instance de classe
FirstChart	Assigne le premier graphique du terminal client à l'instance de classe

© 2000-2024, MetaQuotes Ltd.

Accès aux données protégées	
NextChart	Assigne le graphique suivant du terminal client à l'instance de classe
Open	Ouvre un graphique avec les paramètres spécifiés et l'assigne à l'instance de classe
Detach	Détache le graphique de l'instance de classe
Close	Ferme le graphique assigné à l'instance de classe
BringToTop	Affiche le graphique par-dessus tous les autres graphiques
EventObjectCreate	Définit un flag pour envoyer les notifications d'évènements de création d'un nouvel objet à un graphique
EventObjectDelete	Définit un flag pour envoyer les notificiations d'évènements de suppression d'un objet à un graphique
Indicateurs	
IndicatorAdd	Ajoute un indicateur avec le handler spécifié dans la sous-fenêtre spécifiée du graphique
IndicatorDelete	Supprime un indicateur ayant le nom spécifié de la sous-fenêtre spécifiée du graphique
IndicatorsTotal	Retourne le nombre total d'indicateurs appliqués dans la sous-fenêtre spécifiée du graphique
IndicatorName	Retourne le nom court de l'indicateur de la sous-fenêtre spécifiée du graphique
Navigation	
Navigate	Se déplace dans le graphique
Accès à l'API MQL5	
Symbol	Retourne le symbole affiché dans le graphique
Period	Retourne l'espace de temps du graphique
Redraw	Redessine le graphique assigné à l'instance de classe
GetInteger	Retourne la valeur de la propriété correspondante de l'objet
SetInteger	Définit la nouvelle valeur de la propriété de type integer
GetDouble	Retourne la valeur de la propriété correspondante de l'objet
SetDouble	Définit la nouvelle valeur de la propriété de type double
GetString	Retourne la valeur de la propriété correspondante de l'objet
SetString	Définit la nouvelle valeur de la propriété de type string
SetSymbolPeriod	Change le symbole et l'espace de temps du graphique assigné à l'instance de classe

© 2000-2024, MetaQuotes Ltd.

Accès aux données protégées	
ApplyTemplate	Applique le modèle spécifié au graphique
ScreenShot	Crée une copie d'écran du graphique spécifié et le sauvegarde dans un fichier .gif
WindowOnDropped	Retourne le numéro de la sous-fenêtre correspondant au point de dépose de l'objet (expert ou script)
PriceOnDropped	Retourne le prix correspondant au point de dépose de l'objet (expert ou script)
TimeOnDropped	Retourne la date/heure correspondant au point de dépose de l'objet (expert ou script)
XOnDropped	Retourne la coordonnée X correspondant au point de dépose de l'objet (expert ou script)
YOnDropped	Retourne la coordonnée Y correspondant au point de dépose de l'objet (expert ou script)
Entrée/Sortie	
virtual Save	Sauvegarde les paramètres de l'objet dans un fichier
virtual Load	Charges les paramètres de l'objet depuis un fichier
virtual Type	Retourne l'identifiant du type de l'objet graphique

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

© 2000-2024, MetaQuotes Ltd.

ChartID

Retourne l'identifiant du graphique.

```
long  ChartID() const
```

Valeur de retour

L'identifiant du graphique assigné à l'instance de classe. Si aucun objet n'est assigné, retourne -1.

© 2000-2024, MetaQuotes Ltd.

Mode (Méthode "Get")

Retourne la valeur de la propriété "Mode" (barre, bougie ou ligne)

```
ENUM_CHART_MODE  Mode() const
```

Valeur de retour

Valeur de la propriété "Mode" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Mode (Méthode "Set")

Définit la valeur de la propriété "Mode" (barre, bougie ou ligne)

```
bool  Mode(
   ENUM_CHART_MODE  mode      // nouveau mode du graphique
   )
```

Paramètres

mode

[in] Mode du graphique (bougie, barre ou ligne) de l'énumération ENUM_CHART_MODE.

Valeur de retour

vrai si réalisé avec succès, faux si le mode n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Foreground (Méthode "Get")

Retourne la valeur de la propriété "Foreground".

```
bool  Foreground() const
```

Valeur de retour

Valeur de la propriété "Foreground" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Foreground (Méthode "Set")

Définit une nouvelle valeur à la propriété "Foreground".

```
bool  Foreground(
   bool  foreground      // nouvelle valeur
   )
```

Paramètres

foreground

[in] Nouvelle valeur de la propriété "Foreground".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Shift (Méthode "Get")

Retourne la valeur de la propriété "Shift".

```
bool  Shift() const
```

Valeur de retour

Valeur de la propriété "Shift" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Shift (Méthode "Set")

Définit une nouvelle valeur à la propriété "Shift".

```
bool  Shift(
   bool  shift      // nouvelle valeur du flag
   )
```

Paramètres

shift

[in] Nouvelle valeur pour la propriété "Shift".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShiftSize (Méthode "Get")

Retourne la valeur de la propriété "ShiftSize" (en pourcentage).

```
double  ShiftSize() const
```

Valeur de retour

Valeur de la propriété "ShiftSize" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

ShiftSize (Méthode "Set")

Définit la valeur de la propriété "ShiftSize" (en pourcentage)

```
bool  ShiftSize(
   double  shift_size      // nouvelle valeur de la propriété
   )
```

Paramètres

shift_size

[in] Nouvelle valeur de la propriété "ShiftSize" (en pourcentage).

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

AutoScroll (Méthode "Get")

Retourne la valeur de la propriété "AutoScroll".

```
bool  AutoScroll() const
```

Valeur de retour

Valeur de la propriété "AutoScroll" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

AutoScroll (Méthode "Set")

Définit une nouvelle valeur à la propriété "AutoScroll".

```
bool  AutoScroll(
   bool  autoscroll      // nouvelle valeur du flag
   )
```

Paramètres

autoscroll

[in] Nouvelle valeur pour la propriété "Autoscroll".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

Scale (Méthode "Get")

Retourne la valeur de la propriété "Scale".

```
int  Scale() const
```

Valeur de retour

Valeur de la propriété "Scale" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

Scale (Méthode "Set")

Définit une nouvelle valeur à la propriété "Scale".

```
bool  Scale(
   int  scale      // nouvelle valeur
   )
```

Paramètres

scale

[in] Nouvelle valeur pour la propriété "Scale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ScaleFix (Méthode "Get")

Retourne la valeur de la propriété "ScaleFix" (échelle de graphique fixe ou pas)

```
bool  ScaleFix() const
```

Valeur de retour

Valeur de la propriété "ScaleFix" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

ScaleFix (Méthode "Set")

Définit une nouvelle valeur à la propriété "ScaleFix".

```
bool  ScaleFix(
   bool  scale_fix      // nouvelle valeur
   )
```

Paramètres

scale_fix

[in] Nouvelle valeur pour la propriété "ScaleFix".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ScaleFix_11 (Méthode "Get")

Retourne la valeur de la propriété "ScaleFix_11" (échelle de graphique 1:1, ou autre)

```
bool  ScaleFix_11() const
```

Valeur de retour

Valeur de la propriété "ScaleFix_11" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

ScaleFix_11 (Méthode "Set")

Définit une nouvelle valeur à la propriété "ScaleFix_11".

```
bool  ScaleFix_11(
   string  scale_11      // nouvelle valeur
   )
```

Paramètres

scale_11

[in] Nouvelle valeur pour la propriété "ScaleFix_11".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

FixedMax (Méthode "Get")

Retourne la valeur de la propriété "FixedMax" (prix maximum fixé)

```
double  FixedMax() const
```

Valeur de retour

Valeur de la propriété "FixedMax" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

FixedMax (Méthode "Set")

Définit une nouvelle valeur à la propriété "FixedMax".

```
bool  FixedMax(
   double  max      // nouveau maximum fixé
   )
```

Paramètres

max

[in] Nouvelle valeur pour la propriété "FixedMax".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

FixedMin (Méthode "Get")

Retourne la valeur de la propriété "FixedMin" (prix minimal fixé)

```
double  FixedMin() const
```

Valeur de retour

Valeur de la propriété "Fixed" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

FixedMin (Méthode "Set")

Définit une nouvelle valeur à la propriété "FixedMin".

```
bool  FixedMax(
   double  min      // nouveau minimum fixé
   )
```

Paramètres

max

[in] Nouvelle valeur pour la propriété "FixedMin".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

PointsPerBar (Méthode "Get")

Retourne la valeur de la propriété "PointsPerBar" (en points par barre)

```
double  PointsPerBar() const
```

Valeur de retour

Valeur de la propriété "PointsPerBar" de l'objetassigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

PointsPerBar (Méthode "Set")

Définit une nouvelle valeur à la propriété "PointsPerBar".

```
bool  PointsPerBar(
   double  ppb      // nouvelle échelle (en points par barre)
   )
```

Paramètres

ppb

[in] Nouvelle valeur d'échelle (en points par barre).

Valeur de retour

vrai si réalisé avec succès, faux si l'échelle n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ScalePPB (Méthode "Get")

Retourne la valeur de la propriété "ScalePPB" (l'échelle est "point par barre" ou autre)

```
bool  ScalePPB() const
```

Valeur de retour

Valeur de la propriété "ScalePPB" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

ScalePPB (Méthode "Set")

Définit une nouvelle valeur à la propriété "ScalePPB".

```
bool  ScalePPB(
   bool  scale_ppb      // nouvelle valeur du flag
   )
```

Paramètres

scale_ppb

[in] Nouvelle valeur pour la propriété "ScalePPB".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShowOHLC (Méthode "Get")

Retourne la valeur de la propriété "ShowOHLC".

```
bool  ShowOHLC() const
```

Valeur de retour

Valeur de la propriété "ShowOHLC" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

ShowOHLC (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowOHLC".

```
bool  ShowOHLC(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowOHLC".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShowLineBid (Méthode "Get")

Retourne la valeur de la propriété "ShowLineBid."

```
bool  ShowLineBid() const
```

Valeur de retour

Valeur de la propriété "ShowLineBid" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowLineBid (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowLineBid".

```
bool  ShowLineBid(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowLineBid".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShowLineAsk (Méthode "Get")

Retourne la valeur de la propriété "ShowLineAsk".

```
bool  ShowLineAsk() const
```

Valeur de retour

Valeur de la propriété "ShowLineAsk" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowLineAsk (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowLineAsk".

```
bool  ShowLineAsk(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowLineAsk".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShowLastLine (Méthode "Get")

Retourne la valeur de la propriété "ShowLastLine".

```
bool  ShowLastLine() const
```

Valeur de retour

Valeur de la propriété "ShowLastLine" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowLastLine (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowLastLine".

```
bool  ShowLastLine(
   bool  show      // nouvelle valeur du flag
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowLastLine".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShowPeriodSep (Méthode "Get")

Retourne la valeur de la propriété "ShowPeriodSep" (afficher les séparateurs de période).

```
bool  ShowPeriodSep() const
```

Valeur de retour

Valeur de la propriété "ShowPeriodSep" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowPeriodSep (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowPeriodSep".

```
bool  ShowPeriodSep(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowPeriodSep".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShowGrid (Méthode "Get")

Retourne la valeur de la propriété "ShowGrid".

```
bool  ShowGrid() const
```

Valeur de retour

Valeur de la propriété "ShowGrid" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowGrid (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowGrid".

```
bool  ShowGrid(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowGrid".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShowVolumes (Méthode "Get")

Retourne la valeur de la propriété "ShowVolumes".

```
bool  ShowVolumes() const
```

Valeur de retour

Valeur de la propriété "ShowVolumes" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowVolumes (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowVolumes".

```
bool  ShowVolumes(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowVolumes".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShowObjectDescr (Méthode "Get")

Retourne la valeur de la propriété "ShowObjectDescr" (afficher la description des objets graphiques)

```
bool  ShowObjectDescr() const
```

Valeur de retour

Valeur de la propriété "ShowObjectDescr" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowObjectDescr (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowObjectDescr".

```
bool  ShowObjectDescr(
   bool  show      // Nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowObjectDescr".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShowDateScale

Définit une nouvelle valeur à la propriété "ShowDateScale".

```
bool  ShowDateScale(
   bool  show      // Nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowDateScale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ShowPriceScale

Définit une nouvelle valeur à la propriété "ShowPriceScale".

```
bool  ShowPriceScale(
   bool  show      // Nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowPriceScale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorBackground (Méthode "Get")

Retourne la valeur de la propriété "ColorBackground" (couleur d'arrière plan du graphique).

```
color  ColorBackground() const
```

Valeur de retour

Valeur de la propriété "ColorBackground" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorBackground (Méthode "Set")

Définit la nouvelle valeur de la propriété "ColorBackground".

```
bool  ColorBackground(
   color  new_color      // nouvelle couleur d'arrière plan
   )
```

Paramètres

new_color

[in] Nouvelle couleur d'arrière plan.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorForeground (Méthode "Get")

Retourne la valeur de la propriété "ColorForeground" (couleur de premier plan des axes, de l'échelle et des textes OHLC)

```
color  ColorForeground() const
```

Valeur de retour

Valeur de la propriété "ColorForeground" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorForeground (Méthode "Set")

Définit la valeur de la propriété "ColorForeground" (our les axes, l'échelle et les textes OHLC)

```
bool  ColorForeground(
   color  new_color      // Nouvelle couleur
   )
```

Paramètres

new_color

[in] Nouvelle couleur des axes, de l'échelle et des textes OHLC.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorGrid (Méthode "Get")

Retourne la valeur de la propriété "ColorGrid" (couleur de la grille).

```
color  ColorGrid() const
```

Valeur de retour

Valeur de la propriété "ColorGrid" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorGrid (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorGrid".

```
bool  ColorGrid(
   color  new_color      // nouvelle couleur de grille
   )
```

Paramètres

new_color

[in] Nouvelle couleur de grille.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorBarUp (Méthode "Get")

Retourne la valeur de la propriété "ColorBarUp" (couleur des barres haussières, leur ombre et le contour des corps des bougies).

```
color  ColorBarUp() const
```

Valeur de retour

Valeur de la propriété "ColorBarUp" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorBarUp (Méthode "Set")

Définit la nouvelle valeur de la propriété "ColorBarUp".

```
bool  ColorBarUp(
   color  new_color      // nouvelle couleur des barres haussières
   )
```

Paramètres

new_color

[in] Nouvelle couleur des barres haussières.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorBarDown (Méthode "Get")

Retourne la valeur de la propriété "ColorBarDown" (couleur des barres baissières, leur ombre et le contour du corps des bougies).

```
color  ColorBarDown() const
```

Valeur de retour

Valeur de la propriété "ColorBarDown" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorBarDown (Méthode "Set")

Définit la nouvelle valeur de la propriété "ColorBarDown".

```
bool  ColorBarDown(
   color  new_color      // nouvelle couleur des barres baissières
   )
```

Paramètres

new_color

[in] Nouvelle couleur des barres baissières.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorCandleBull (Méthode "Get")

Retourne la valeur de la propriété "ColorCandleBull" (couleur du corps des bougies haussières).

```
color  ColorCandleBull() const
```

Valeur de retour

Valeur de la propriété "ColorCandleBull" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorCandleBull (Méthode "Set")

Définit une nouvelle valeur pour la propriété "ColorCandleBull".

```
bool  ColorCandleBull(
   color  new_color      // nouvelle couleur du corps des bougies haussières
   )
```

Paramètres

new_color

[in] Nouvelle couleur du corps des bougies haussières.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorCandleBear (Méthode "Get")

Retourne la valeur de la propriété "ColorCandleBear" (couleur du corps des bougies baissières).

```
color  ColorCandleBear() const
```

Valeur de retour

Valeur de la propriété "ColorCandleBear" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorCandleBear (Méthode "Set")

Définit une nouvelle valeur pour la propriété "ColorCandleBear".

```
bool  ColorCandleBear(
   color  new_color      // nouvelle couleur du corps des bougies baissières
   )
```

Paramètres

new_color

[in] Nouvelle couleur du corps des bougies baissières.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorChartLine (Méthode "Get")

Retourne la valeur de la propriété "ColorChartLine" (couleur des graphiques en ligne et des bougies en Doji)

```
color  ColorChartLine() const
```

Valeur de retour

Valeur de la propriété "ColorChartLine" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorChartLine (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorChartLine".

```
bool  ColorChartLine(
   color  new_color      // nouvelle couleur des graphiques en ligne
   )
```

Paramètres

new_color

[in] Nouvelle couleur des graphiques en ligne (et des bougies en Doji).

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorVolumes (Méthode "Get")

Retourne la valeur de la propriété "ColorVolumes" (couleur des volumes et des niveaux des positions ouvertes)

```
color  ColorVolumes() const
```

Valeur de retour

Valeur de la propriété "ColorVolumes" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorVolumes (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorVolumes".

```
bool  ColorVolumes(
   color  new_color      // nouvelle couleur des volumes (niveaux des positions ouvert
   )
```

Paramètres

new_color

[in] Nouvelle couleur des volumes (niveaux des positions ouvertes).

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorLineBid (Méthode "Get")

Retourne la valeur de la propriété "ColorLineBid" (couleur de la ligne Bid)

```
color  ColorLineBid() const
```

Valeur de retour

Valeur de la propriété "ColorLineBid" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorLineBid (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorLineBid".

```
bool  ColorLineBid(
   color  new_color      // nouvelle couleur de la ligne Bid
   )
```

Paramètres

new_color

[in] Nouvelle couleur de la ligne Bid.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorLineAsk (Méthode "Get")

Retourne la valeur de la propriété "ColorLineAsk" (couleur de la ligne Ask)

```
color  ColorLineAsk() const
```

Valeur de retour

Valeur de la propriété "ColorLineAsk" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorLineAsk (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorLineAsk".

```
bool  ColorLineAsk(
   color  new_color      // nouvelle couleur de la ligne Ask
   )
```

Paramètres

new_color

[in] Nouvelle couleur de la ligne Ask.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorLineLast (Méthode "Get")

Retourne la valeur de la propriété "ColorLineLast" (couleur de la ligne du dernier deal)

```
color  ColorLineLast() const
```

Valeur de retour

Valeur de la propriété "ColorLineLast" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorLineLast (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorLineLast".

```
bool  ColorLineLast(
   color  new_color      // nouvelle couleur de la ligne du dernier deal
   )
```

Paramètres

new_color

[in] Nouvelle couleur de la ligne du dernier deal.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ColorStopLevels (Méthode "Get")

Retourne la valeur de la propriété "ColorStopLevels" (couleur des niveaux de SL et TP)

```
color  ColorStopLevels() const
```

Valeur de retour

Valeur de la propriété "ColorStopLevels" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorStopLevels (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorStopLevels".

```
bool  ColorStopLevels(
   color  new_color      // nouvelle couleur des niveaux de SL et TP
   )
```

Paramètres

new_color

[in] nouvelle couleur des niveaux de Stop Loss et Prix du Take Profit

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

VisibleBars

Retourne le nombre total de barres visibles sur le graphique.

```
int  VisibleBars() const
```

Valeur de retour

Retourne le nombre total de barres visibles sur le graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

WindowsTotal

Retourne le nombre total de fenêtres de graphiques, incluant les sous-fenêtres des indicateurs.

```
int  WindowsTotal() const
```

Valeur de retour

Nombre total de fenêtres, incluant les sous-fenêtres des indicateurs, assignées à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

WindowIsVisible

Retourne le flag de visibilité de la sous-fenêtre spécifiée du graphique.

```
bool  WindowIsVisible(
   int  num      // numéro de la sous-fenêtre
   ) const
```

Paramètres

num

[in] Numéro de la sous-fenêtre (0 signifie la fenêtre principale).

Valeur de retour

Retourne le flag de visibilité de la sous-fenêtre spécifiée du graphique assigné à l'instance du graphique. Si aucun graphique n'est assigné, retourne faux.

© 2000-2024, MetaQuotes Ltd.

WindowHandle

Retourne le handler de la fenêtre du graphique (HWND).

```
int  WindowHandle() const
```

Valeur de retour

Handler de la fenêtre du graphique assigné à l'instance du graphique. Si aucun graphique n'est assigné, retourne INVALID_HANDLE.

© 2000-2024, MetaQuotes Ltd.

FirstVisibleBar

Retourne le numéro de la première barre visible du graphique.

```
int  FirstVisibleBar() const
```

Valeur de retour

Numéro de la première barre visible du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne -1.

© 2000-2024, MetaQuotes Ltd.

WidthInBars

Retourne la largeur de la fenêtre en nombre de barres.

```
int  WidthInBars() const
```

Valeur de retour

Largeur de la fenêtre en barres assignée à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

WidthInPixels

Retourne la largeur de la sous-fenêtre en pixels.

```
int  WidthInPixels() const
```

Valeur de retour

Largeur en pixels de la sous-fenêtre du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

HeightInPixels

Retourne la hauteur de la sous-fenêtre en pixels.

```
int  HeightInPixels(
   int  num      // numéro de la sous-fenêtre
   ) const
```

Paramètres

num

[in] Numéro de la sous-fenêtre (0 signifie la fenêtre principale).

Valeur de retour

La hauteur de la sous-fenêtre du graphique en pixels. Si aucun graphique n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

PriceMin

Retourne le prix minimum de la sous-fenêtre spécifiée.

```
double  PriceMin(
   int  num      // numéro de la sous-fenêtre
   ) const
```

Paramètres

num

[in] Numéro de la sous-fenêtre (0 signifie la fenêtre principale).

Valeur de retour

Prix minimal du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne EMPTY_VALUE.

© 2000-2024, MetaQuotes Ltd.

PriceMax

Retourne le prix maximum de la sous-fenêtre spécifiée.

```
double  PriceMax(
   int  num      // numéro de la sous-fenêtre
   ) const
```

Paramètres

num

[in] Numéro de la sous-fenêtre (0 signifie la fenêtre principale).

Valeur de retour

Prix maximal du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne EMPTY_VALUE.

© 2000-2024, MetaQuotes Ltd.

Attach

Assigne le graphique actuel à l'instance de classe.

```
void  Attach()
```

Attach

Assigne le graphique spécifié à l'instance de classe.

```
void  Attach(
   long  chart       // Identifiant du graphique
   )
```

Paramètres

chart

[in] Identifiant du graphique à assigner.

© 2000-2024, MetaQuotes Ltd.

FirstChart

Assigne le premier graphique du terminal client à l'instance de classe

```
void  FirstChart()
```

© 2000-2024, MetaQuotes Ltd.

NextChart

Assigne le graphique suivant du terminal client à l'instance de classe.

```
void  NextChart()
```

© 2000-2024, MetaQuotes Ltd.

Open

Ouvre le graphique avec les paramètres spécifiés et l'assigne à l'instance de classe.

```
long  Open(
   const string     symbol_name,      // Nom du symbole
   ENUM_TIMEFRAMES  timeframe         // Période
   )
```

Paramètres

symbol_name

[in] Nom du symbole. NULL signifie le symbole du graphique courant (auquel l'expert est attaché).

timeframe

[in] Espace de temps du graphique (énumération ENUM_TIMEFRAMES). 0 signifie l'espace de temps actuel.

Valeur de retour

Identifiant du graphique.

© 2000-2024, MetaQuotes Ltd.

Detach

Détache le graphique de l'instance de classe

```
void  Detach()
```

© 2000-2024, MetaQuotes Ltd.

Close

Ferme le graphique assigné à l'instance de classe.

```
void  Close()
```

© 2000-2024, MetaQuotes Ltd.

BringToTop

Affiche le graphique par-dessus tous les autres graphiques.

```
bool  BringToTop() const
```

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

EventObjectCreate

Définit un flag pour envoyer les notifications d'un évènement de création d'un nouvel objet à tous les programmes MQL5 d'un graphique.

```
bool  EventObjectCreate(
   bool  flag      // flag
   )
```

Paramètres

flag

[in] Nouvelle valeur du flag.

Valeur de retour

vrai si réalisé avec succès, faux si le flag n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

EventObjectDelete

Définit un flag pour envoyer les notifications d'un évènement de suppression d'un objet à tous les programmes MQL5 d'un graphique.

```
bool  EventObjectDelete(
   bool  flag      // flag
   )
```

Paramètres

flag

[in] Nouvelle valeur du flag.

Valeur de retour

vrai si réalisé avec succès, faux si le flag n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

IndicatorAdd

Ajoute un indicateur avec le handler spécifié dans la sous-fenêtre spécifiée du graphique

```
bool  IndicatorAdd(
   int   sub_win         // numéro de la sous-fenêtre
   int   handle          // handle de l'indicateur
   );
```

Paramètres

sub_win

[in] Numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre du graphique principal. Si le numéro de sous-fenêtre spécifié n'existe pas, une nouvelle fenêtre est créée.

handle

[in] Handle de l'indicateur.

Valeur de Retour

La fonction retourne vrai en cas de succès, sinon retourne faux. Pour obtenir plus d'information concernant l'error, appeler la fonction GetLastError().

Voir aussi

IndicatorDelete(), IndicatorsTotal(), IndicatorName().

© 2000-2024, MetaQuotes Ltd.

IndicatorDelete

Supprime un indicateur ayant le nom spécifié dans la fenêtre spécifiée du graphique.

```
bool  IndicatorDelete(
   int            sub_win       // numéro de la sous-fenêtre
   const string   name          // nom court de l'indicateur
   );
```

Paramètres

sub_win

[in] Numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre du graphique principal.

const name

[in] Le nom court de l'indicateur qui est spécifié dans la propriété INDICATOR_SHORTNAME avec la fonction IndicatorSetString(). Pour obtenir le nom court d'un indicateur, utiliser la fonction IndicatorName() .

Valeur de Retour

Retourne vrai si l'indicateur a été supprimé avec succès. Sinon retourne faux. Pour obtenir les détails de l'erreur, utilisez la fonction GetLastError().

Note

Si deux indicateurs avec des noms courts identiques existent dans la sous-fenêtre du graphique, le premier sera supprimé.

Si d'autres indicateurs sur ce graphique sont basés sur les valeurs de l'indicateur en cours de suppression, ces indicateurs seront supprimés.

Ne pas confondre le nom court de l'indicateur et le nom du fichier spécifié lors de la création d'un indicateur en utilisant les fonctions iCustom() et IndicatorCreate(). Si le nom court d'un indicateur n'est pas défini explicitement, alors le nom du fichier contenant le code source de l'indicateur sera défini pendant la compilation.

La suppression d'un indicateur d'un graphique ne signifie pas que les calculs seront supprimés de la mémoire du terminal. Pour relâcher le handle de l'indicateur, utiliser la fonction IndicatorRelease().

Le nom court de l'indicateur doit être formé correctement. Il sera affecté à la propriété INDICATOR_SHORTNAME avec la fonction IndicatorSetString(). Il est recommendé que le nom court contienne tous les paramètres d'entrée de l'indicateur, car l'indicateur à supprimer du graphique par la fonction IndicatorDelete() est identifié par son nom court.

Voir aussi

IndicatorAdd(), IndicatorsTotal(), IndicatorName(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2024, MetaQuotes Ltd.

IndicatorsTotal

Retourne le nombre total d'indicateurs appliqués à la fenêtre du graphique spécifié.

```
int  IndicatorsTotal(
   int   sub_win     // numéro de la sous-fenêtre
   );
```

Paramètres

sub_win

[in] Numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre du graphique principal.

Valeur de Retour

Le nombre d'indicateurs dans la fenêtre spécifiée. Pour obtenir les détails de l'erreur, utilisez la fonction GetLastError().

Note

La fonction permet de rechercher parmi tous les indicateurs attachés au graphique. Le nombre total de fenêtres du graphique peut être obtenu depuis la propriété CHART_WINDOWS_TOTAL en utilisant la fonction GetInteger().

Voir aussi

IndicatorAdd(), IndicatorDelete(), IndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2024, MetaQuotes Ltd.

IndicatorName

Retourne le nom court de l'indicateur par son index dans la liste des indicateurs de la fenêtre du graphique spécifié.

```
string  IndicatorName(
   int   sub_win      // numéro de la sous-fenêtre
   int   index        // index de l'indicateur dans la liste des indicateurs ajoutés à
   );
```

Paramètres

sub_win

[in] Numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre du graphique principal.

index

[in] index de l'indicateur dans la liste des indicateurs. L'énumeration des indicateurs commence à zero, i.e. le premier indicateur de la liste à l'index 0. Pour obtenir le nombre d'indicateurs dans la liste, utiliser la fonction IndicatorsTotal().

Valeur de Retour

Le nom court de l'indicateur qui est affecté à la propriété INDICATOR_SHORTNAME avec la fonction IndicatorSetString(). Pour obtenir les détails de l'erreur, utiliser la fonction GetLastError().

Note

Ne pas confondre le nom court de l'indicateur et le nom du fichier spécifié lors de la création d'un indicateur en utilisant les fonctions iCustom() et IndicatorCreate(). Si le nom court d'un indicateur n'est pas défini explicitement, alors le nom du fichier contenant le code source de l'indicateur sera défini pendant la compilation.

Le nom court de l'indicateur doit être formé correctement. Il sera affecté à la propriété INDICATOR_SHORTNAME avec la fonction IndicatorSetString(). Il est recommendé que le nom court contienne tous les paramètres d'entrée de l'indicateur, car l'indicateur à supprimer du graphique par la fonction IndicatorDelete() est identifié par son nom court.

Voir aussi

IndicatorAdd(), IndicatorDelete, IndicatorsTotal, iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2024, MetaQuotes Ltd.

Navigate

Permet de se déplacer dans le graphique.

```
bool  Navigate(
   ENUM_CHART_POSITION  position,      // Position
   int                  shift=0        // Décalage
   )
```

Paramètres

position

[in] Valeur de l'énumération ENUM_CHART_POSITION.

shift=0

[in] Nombre de barres de décalage.

Valeur de retour

vrai si réalisé avec succès, faux si le graphique n'a pas été déplacé.

© 2000-2024, MetaQuotes Ltd.

Symbol

Retourne le symbole du graphique.

```
string  Symbol() const
```

Valeur de retour

Symbole du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

Period

Retourne l'espace de temps du graphique.

```
ENUM_TIMEFRAMES  Period() const
```

Valeur de retour

L'espace de temps du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2024, MetaQuotes Ltd.

Redraw

Redessine le graphique assigné à l'instance de classe.

```
void  Redraw()
```

© 2000-2024, MetaQuotes Ltd.

GetInteger

Cette fonction retourne la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être de type integer (entier). Cette fonction possède 2 variantes.

1. Retourne immédiatement la valeur de la propriété.

```
long  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,          // identifiant de la propriété
   int                          sub_window=0      // numéro de la sous-fenêtre
   ) const
```

2. En cas de succès, affecte la valeur de la propriété dans la variable de type integer spécifiée et passée par référence comme dernier paramètre.

```
bool  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,         // identifiant de la propriété
   int                          sub_window,      // numéro de la sous-fenêtre
   long&                        value            // la valeur est récupérée ici
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumeration ENUM_CHART_PROPERTY_INTEGER).

sub_window

[in] Numéro de la sous-fenêtre du graphique.

value

[in] Variable de type integer qui reçoit la valeur de la propriété demandée.

Valeur de Retour

Valeur de la propriété du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne -1.

Pour la seconde variante de la fonction, retourne vrai si cette propriété est trouvée et que la valeur a été copiée dans la variable, sinon retourne faux. Pour en savoir plus sur les erreurs, utiliser la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

SetInteger

Définit la nouvelle valeur de la propriété de type integer.

```
bool  SetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,      // identifiant de la propriété
   long                         value         // nouvelle valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumeration ENUM_CHART_PROPERTY_INTEGER).

value

[in] Nouvelle valeur de la propriété.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété de type integer n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

GetDouble

Cette fonction retourne la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être de type double. Cette fonction possède 2 variantes.

1. Retourne immédiatement la valeur de la propriété.

```
double  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // identifiant de la propriété
   int                         sub_window=0     // numéro de la sous-fenêtre
   ) const
```

2. En cas de succès, affecte la valeur de la propriété dans la variable de type double spécifiée et passée par référence comme dernier paramètre.

```
bool  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,        // identifiant de la propriété
   int                         sub_window,     // numéro de la sous-fenêtre
   double&                     value           // la valeur est récupérée ici
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumération ENUM_CHART_PROPERTY_DOUBLE).

sub_window

[in] Numéro de la sous-fenêtre du graphique.

value

[in] Variable de type double qui reçoit la valeur de la propriété demandée.

Valeur de Retour

Valeur de la propriété du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne EMPTY_VALUE.

Pour la seconde variante de la fonction, retourne vrai si cette propriété est trouvée et que la valeur a été copiée dans la variable, sinon retourne faux. Pour en savoir plus sur les erreurs, utiliser la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

SetDouble

Définit la nouvelle valeur de la propriété de type double.

```
bool  SetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,      // identifiant de la propriété
   double                      value         // nouvelle valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumération ENUM_CHART_PROPERTY_DOUBLE).

value

[in] Nouvelle valeur pour la propriété.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété de type double n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

GetString

Cette fonction retourne la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être de type string. Cette fonction possède 2 variantes.

1. Retourne immédiatement la valeur de la propriété.

```
string  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id      // identifiant de la propriété
   ) const
```

2. En cas de succès, affecte la valeur de la propriété dans la variable de type string spécifiée et passée par référence comme dernier paramètre.

```
bool  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,     // identifiant de la propriété
   string&                     value        // la valeur est récupérée ici
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumération ENUM_CHART_PROPERTY_STRING).

sub_window

[in] Numéro de la sous-fenêtre du graphique.

value

[in] Variable de type string qui reçoit la valeur de la propriété demandée.

Valeur de Retour

Valeur de la propriété du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne "".

Pour la seconde variante de la fonction, retourne vrai si cette propriété est trouvée et que la valeur a été copiée dans la variable, sinon retourne faux. Pour en savoir plus sur les erreurs, utiliser la fonction GetLastError().

© 2000-2024, MetaQuotes Ltd.

SetString

Définit la nouvelle valeur de la propriété de type string.

```
bool  SetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,      // identifiant de la propriété
   string                      value         // nouvelle valeur de la propriété
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumération ENUM_CHART_PROPERTY_STRING).

value

[in] Nouvelle valeur pour la propriété.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété de type string n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Change le symbole et l'espace de temps du graphique assignés à l'instance de classe.

```
bool  SetSymbolPeriod(
   const string     symbol_name,      // symbole
   ENUM_TIMEFRAMES  timeframe         // Période
   )
```

Paramètres

symbol_name

[in] Nom du nouveau symbole. NULL signifie le symbole du graphique courant (auquel l'expert est attaché).

timeframe

[in] Nouvelle espace de temps du graphique (énumération ENUM_TIMEFRAMES). 0 signifie l'espace de temps actuel.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

ApplyTemplate

Applique le modèle spécifié au graphique.

```
bool  ApplyTemplate(
   const string  filename      // nom du fichier modèle
   )
```

Paramètres

filename

[in] Nom du fichier du modèle.

Valeur de retour

vrai si réalisé avec succès, faux si le modèle n'a pas été appliqué.

© 2000-2024, MetaQuotes Ltd.

ScreenShot

Crée une copie d'écran du graphique spécifié et le sauvegarde dans un fichier .gif.

```
bool  ScreenShot(
   string           filename,                  // Nom de fichier
   int              width,                     // Largeur
   int              height,                    // Hauteur
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // Type d'alignement
   ) const
```

Paramètres

filename

[in] Nom du fichier destination de la copie d'écran.

width

[in] Largeur de la copie d'écran en pixels.

height

[in] Hauteur de la copie d'écran en pixels.

align_mode=ALIGN_RIGHT

[in] Mode d'alignement si la copie d'écran n'est pas assez large.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

WindowOnDropped

Retourne le numéro de la sous-fenêtre du graphique correspondant au point de dépose de l'objet (expert ou script).

```
int  WindowOnDropped() const
```

Valeur de retour

Numéro de la sous-fenêtre du graphique correspondant au point de dépose de l'objet. 0 signifie la fenêtre du graphique principal.

© 2000-2024, MetaQuotes Ltd.

PriceOnDropped

Retourne le prix correspondant au point de dépose de l'objet (expert ou script).

```
double  PriceOnDropped() const
```

Valeur de retour

Le prix correspondant au point de dépose de l'objet.

© 2000-2024, MetaQuotes Ltd.

TimeOnDropped

Retourne la date/heure correspondant au point de dépose de l'objet (expert ou script).

```
datetime  TimeOnDropped() const
```

Valeur de retour

Date/heure correspondant au point de dépose de l'objet.

© 2000-2024, MetaQuotes Ltd.

XOnDropped

Retourne la coordonnée X correspondant au point de dépose de l'objet (expert ou script).

```
int  XOnDropped() const
```

Valeur de retour

Coordonnée X du point de dépose de l'objet.

© 2000-2024, MetaQuotes Ltd.

YOnDropped

Retourne la coordonnée Y correspondant au point de dépose de l'objet (expert ou script).

```
int  YOnDropped() const
```

Valeur de retour

Coordonnée Y du point de dépose de l'objet.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire déjà ouvert avec la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire déjà ouvert avec la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet graphique.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type d'objet (0x1111 pour CChart).

© 2000-2024, MetaQuotes Ltd.

Graphiques

La bibliothèque des graphiques contient des classes et des fonctions globales pour le dessin rapide de graphiques personnalisés. La bibliothèque fournit des solutions toutes prêtes et pratiques pour construire des axes, des courbes, mais fournit aussi des méthodes pour changer rapidement les propriétés communes d'un graphique personnalisé.

Pour commencer à travailler avec la bibliothèque, lisez simplement l'article Visualisez ceci ! Bibliothèque graphique MQL5 similaire au 'plot' du langage R.

La bibliothèque des graphiques est située dans le répertoire Include\Graphics du répertoire de travail du terminal.

Classe	Description
CAxis	Classe pour travailler les axes de coordonnées
ColorGenerator	Classz spécifiant le schéma de couleurs par défaut
CCurve	Classe pour travailler avec les courbes
CGraphic	Classe de base pour la créations de graphiques personnalisés

© 2000-2024, MetaQuotes Ltd.

GraphPlot

Fonctions pour dessiner rapidement une courbe.

Version dessinant une courbe simple utilisant les coordonnées Y.

```
string  GraphPlot(
   const double      &y[],                  // coordonnées Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // type de courbe
   )
```

Note

Les indices du tableau y sont utilisés comme coordonnées X pour la courbe.

Version dessinant une courbe simple utilisant les coordonnées X et Y

```
string  GraphPlot(
   const double      &x[],                  // coordonnées X
   const double      &y[],                  // coordonnées Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // type de courbe
   )
```

Version dessinant deux courbes utilisant les coordonnées X et Y

```
string  GraphPlot(
   const double      &x1[],                 // coordonnées X
   const double      &y1[],                 // coordonnées Y
   const double      &x2[],                 // coordonnées X
   const double      &y2[],                 // coordonnées Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // type de courbe
   )
```

Version dessinant trois courbes utilisant les coordonnées X et Y

```
string  GraphPlot(
   const double      &x1[],                 // coordonnées X
   const double      &y1[],                 // coordonnées Y
   const double      &x2[],                 // coordonnées X
   const double      &y2[],                 // coordonnées Y
   const double      &x3[],                 // coordonnées X
   const double      &y3[],                 // coordonnées Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // type de courbe
   )
```

© 2000-2024, MetaQuotes Ltd.

Version dessinant une courbe utilisant les coordonnées de points CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points[],            // coordonnées de la courbe
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // type de courbe
   )
```

Version dessinant deux courbes utilisant les coordonnées de points CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points1[],           // coordonnées de la courbe
   const CPoint2D   &points2[],           // coordonnées de la courbe
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // type de courbe
   )
```

Version dessinant trois courbes utilisant les coordonnées de points CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points1[],           // coordonnées de la courbe
   const CPoint2D   &points2[],           // coordonnées de la courbe
   const CPoint2D   &points3[],           // coordonnées de la courbe
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // type de courbe
   )
```

Version dessinant une courbe utilisant un pointeur vers CurveFunction

```
string  GraphPlot(
   CurveFunction    function,             // pointeur vers la fonction
   const double     from,                 // valeur initiale de l'argument
   const double     to,                   // valeur finale de l'argument
   const double     step,                 // incrément de l'argument
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // type de courbe
   )
```

Version dessinant deux courbes utilisant des pointeurs vers des fonctions CurveFunction

```
string  GraphPlot(
   CurveFunction    function1,            // pointeur vers la fonction
   CurveFunction    function2,            // pointeur vers la fonction
   const double     from,                 // valeur initiale de l'argument
   const double     to,                   // valeur finale de l'argument
   const double     step,                 // incrément de l'argument
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // type de courbe
   )
```

© 2000-2024, MetaQuotes Ltd.

Version dessinant trois courbes utilisant des pointeurs vers des fonctions CurveFunction

```
string  GraphPlot(
   CurveFunction    function1,            // pointeur vers la fonction
   CurveFunction    function2,            // pointeur vers la fonction
   CurveFunction    function3,            // pointeur vers la fonction
   const double     from,                 // valeur initiale de l'argument
   const double     to,                   // valeur finale de l'argument
   const double     step,                 // incrément de l'argument
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // type de courbe
   )
```

Paramètres

&x[]

[in] Coordonnées X.

&y[]

[in] Coordonnées Y.

&x1[]

[in] Coordonnées X de la première courbe.

&y1[]

[in] Coordonnées Y de la première courbe.

&x2[]

[in] Coordonnées X de la seconde courbe.

&y2[]

[in] Coordonnées Y de la seconde courbe.

&x3[]

[in] Coordonnées X de la troisième courbe.

&y3[]

[in] Coordonnées Y de la troisième courbe.

&points[]

[in] Coordonnées des points de la courbe.

&points1[]

[in] Coordonnées des points de la première courbe.

&points2[]

[in] Coordonnées des points de la seconde courbe.

&points3[]

[in] Coordonnées des points de la troisième courbe.

© 2000-2024, MetaQuotes Ltd.

function

[in] Pointeur vers la fonction CurveFunction.

function1

[in] Pointeur vers la première fonction.

function2

[in] Pointeur vers la seconde fonction.

function3

[in] Pointeur vers la troisième fonction.

from

[in] Correspond à la première coordonnée X.

to

[in] Correspond à la dernière coordonnée X.

step

[in] Paramètre pour le calcul des coordonnées X.

type=CURVE_POINTS

[in] Type de courbe.

Valeur de Retour

Le nom d'une ressource graphique.

© 2000-2024, MetaQuotes Ltd.

CAxis

CAxis is an auxiliary graphics library class for working with the coordinate axes.

Description

La classe CAxis reçoit et stocke les différents paramètres des axes de coordonnées. La classe implémente la possibilité de mettre à l'échelle automatiquement les axes des coordonnées.

Déclaration

```
class CAxis
```

Titre

```
#include <Graphics\Axis.mqh>
```

Méthodes de la classe

Méthode	Description
AutoScale	Retourne/définit le flag de mise à l'échelle automatique
Min	Retourne/définit la valeur minimum de l'axe
Max	Retourne/définit la valeur maximum de l'axe
Step	Retourne la valeur du pas de l'axe
Name	Retourne/définit le nom de l'axe
Color	Retourne/définit la couleur de l'axe
ValuesSize	Retourne/définit la taille des numéros sur l'axe
ValuesWidth	Retourne/définit la longueur affichée maximum des numéros de l'axe
ValuesFormat	Retourne/définit le format des numéros de l'axe
ValuesDateTimeMode	Retourne le format de conversion d'une date en chaîne de caractères.
ValuesFunctionFormat	Retourne un pointeur vers la fonction définissant le format d'affichage des valeurs sur l'axe.
ValuesFunctionFormatCBData	Retourne un pointeur sur l'objet pouvant contenir des données supplémentaires sur la conversion des valeurs de l'axe.
NameSize	Retourne/définit la taille de la police de caractères du nom de l'axe
ZeroLever	Retourne/définit la valeur du "niveau zéro"

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
DefaultStep	Retourne/définit la valeur initiale du pas de l'axe
MaxLabels	Retourne/définit la quantité maximum de numéros sur l'axe
MinGrace	Retourne/définit la valeur de "tolérance" pour le minimum de l'axe
MaxGrace	Retourne/définit la valeur de "tolérance" pour le maximum de l'axe
SelectAxisScale	Met automatiquement à l'échelle l'axe.

© 2000-2024, MetaQuotes Ltd.

AutoScale (méthode Get)

Retourne le flag définissant le besoin de mise à l'échelle automatique.

```
bool  AutoScale()
```

Valeur de Retour

La valeur du flag.

Note

true — effectue la mise à l'échelle automatique.

false — n'effectue pas la mise à l'échelle automatique.

AutoScale (méthode Set)

Définit le flag définissant le besoin de mise à l'échelle automatique.

```
void  AutoScale(
   const bool  auto      // valeur du flag
   )
```

Paramètres

auto

[in]

Note

true — effectue la mise à l'échelle automatique.

false — n'effectue pas la mise à l'échelle automatique.

© 2000-2024, MetaQuotes Ltd.

Min (méthode Get)

Retourne la valeur minimum de l'axe.

```
double  Min()
```

Valeur de Retour

La valeur minimum de l'axee.

Min (méthode Set)

Définit la valeur minimum de l'axe.

```
void  Min(
   const double  min      // valeur minimum
   )
```

Paramètres

min

[in] Valeur minimum.

© 2000-2024, MetaQuotes Ltd.

Max (méthode Get)

Retourne la valeur maximum de l'axe.

```
double  Max()
```

Valeur de Retour

La valeur maximum de l'axe.

Max (méthode Set)

Définit la valeur maximum de l'axe.

```
void  Max(
   const double  max      // valeur maximum
   )
```

Paramètres

max

[in] Valeur maximum de l'axe.

© 2000-2024, MetaQuotes Ltd.

Step (méthode Get)

Retourne la valeur du pas par axe.

```
double  Step()
```

Valeur de Retour

Valeur de pas.

© 2000-2024, MetaQuotes Ltd.

Name (méthode Get)

Retourne le nom de l'axe.

```
string  Name()
```

Valeur de Retour

Le nom de l'axe.

Name (méthode Set)

Définit le nom de l'axe.

```
void  Name(
   const string  name      // nom de l'axe
   )
```

Paramètres

name

[in] Nom de l'axe.

© 2000-2024, MetaQuotes Ltd.

Color (méthode Get)

Retourne la couleur de l'axe.

```
color  Color()
```

Valeur de Retour

La couleur de l'axe.

Color (méthode Set)

Définit la couleur de l'axe.

```
void  Color(
   const color  clr      // couleur de l'axe
   )
```

Paramètres

clr

[in] Couleur de l'axe.

© 2000-2024, MetaQuotes Ltd.

ValuesSize (méthode Get)

Retourne la taille des numéros de l'axe.

```
int  ValuesSize()
```

Valeur de Retour

La taille des numéros de l'axe.

ValuesSize (méthode Set)

Définit la taille des numéros de l'axe.

```
void  ValuesSize(
   const int  size      // la taille des numéros de l'axe
   )
```

Paramètres

size

[in] La taille des numéros de l'axe

© 2000-2024, MetaQuotes Ltd.

ValuesWidth (méthode Get)

Retourne la longueur maximale autorisée en pixels pour l'affichage des numéros des axes.

```
int  ValuesWidth()
```

Valeur de Retour

La longueur des numéros des axes en pixels.

Note

Si la longueur en pixels d'un nombre spécifié est supérieure à la longueur d'affichage maximum autorisée, il sera tronqué et sera terminé par des points.

ValuesWidth (méthode Set)

Définit la longueur maximale autorisée en pixels pour l'affichage des numéros des axes.

```
void  ValuesWidth(
   const int  width      // longueur maximum autorisée en pixels
   )
```

Paramètres

width

[in] Longueur maximum autorisée pour les numéros des axes.

Note

Si la longueur en pixels d'un nombre spécifié est supérieure à la longueur d'affichage maximum autorisée, il sera tronqué et sera terminé par des points.

© 2000-2024, MetaQuotes Ltd.

ValuesFormat (méthode Get)

Retourne le format des nombres de l'axe.

```
string  ValuesFormat()
```

Valeur de Retour

Le format des nombres.

ValuesFormat (méthode Set)

Définit le format des nombres de l'axe.

```
void  ValuesFormat(
   const string  format      // le format des nombres de l'axe
   )
```

Paramètres

format

[in] Le format des nombres de l'axe.

© 2000-2024, MetaQuotes Ltd.

ValuesDateTimeMode (méthode Get)

Retourne le format de conversion d'une date en chaîne de caractères.

```
int  ValuesDateTimeMode()
```

Valeur de Retour

Format de conversion d'une date en chaîne de caractères.

ValuesDateTimeMode (méthode Set)

Définit le format de conversion d'une date en chaîne de caractères.

```
void  ValuesDateTimeMode(
   const int  mode      // format de conversion d'une date en chaîne de caractères
   )
```

Paramètres

mode

[in] Format de conversion.

Note

Vous trouverez plus d'informations sur les formats de conversion d'une date en une chaîne de caractères dans la description de la fonction TimeToString().

© 2000-2024, MetaQuotes Ltd.

ValuesFunctionFormat (méthode Get)

Retourne un pointeur vers la fonction définissant le format d'affichage des valeurs sur l'axe.

```
DoubleToStringFunction  ValuesFunctionFormat()
```

Valeur de Retour

Pointeur vers la fonction définissant le format d'affichage des valeurs sur l'axe.

ValuesFunctionFormat (méthode Set)

Définit un pointeur vers la fonction définissant le format d'affichage des valeurs sur l'axe.

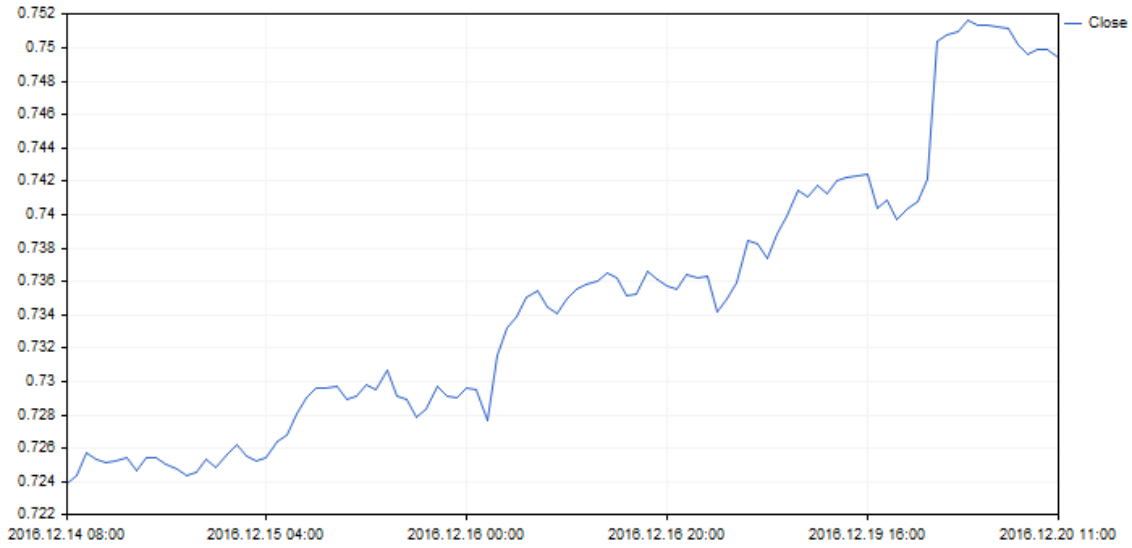
```
void  ValuesFunctionFormat(
   DoubleToStringFunction  func      // fonction pour convertir les valeurs numériques
   )
```

Paramètres

func

[in] Fonction personnalisée pour convertir les valeurs numériques en chaînes de caractères.

Exemple :



Le format d'affichage des valeurs de l'axe X a été changé avec le code suivant :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              DateAxisGraphic.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//--- tableau pour stocker les valeurs
double arrX[];
double arrY[];
//+------------------------------------------------------------------+
//| Fonction personnalisée pour créer les valeurs sur l'axe X        |
//+------------------------------------------------------------------+
string TimeFormat(double x,void *cbdata)
  {
   return(TimeToString((datetime)arrX[ArraySize(arrX)-(int)x-1]));
  }
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlRates rates[];
   CopyRates(Symbol(),Period(),0,100,rates);
   ArraySetAsSeries(rates,true);
   int size=ArraySize(rates);
   ArrayResize(arrX,size);
   ArrayResize(arrY,size);
   for(int i=0; i<size;++i)
     {
      arrX[i]=(double)rates[i].time;
      arrY[i]=rates[i].close;
     }
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"DateAxisGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"DateAxisGraphic");
     }
//--- crée la courbe
   CCurve *curve=graphic.CurveAdd(arrY,CURVE_LINES);
//--- récupère l'axe X
   CAxis *xAxis=graphic.XAxis();
//--- définit les propriétés de l'axe X
   xAxis.AutoScale(false);
   xAxis.Type(AXIS_TYPE_CUSTOM);
   xAxis.ValuesFunctionFormat(TimeFormat);
   xAxis.DefaultStep(20.0);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2024, MetaQuotes Ltd.

ValuesFunctionFormatCBData (méthode Get)

Retourne un pointeur sur l'objet pouvant contenir des données supplémentaires sur la conversion des valeurs de l'axe.

```
void*  ValuesFunctionFormatCBData()
```

Valeur de Retour

Un pointeur vers l'objet pouvant contenir des données supplémentaires sur la conversion des valeurs de l'axe.

ValuesFunctionFormatCBData (méthode Set)

Définit le pointeur vers un objet pouvant contenir des données supplémentaires sur la conversion des valeurs de l'axe.

```
void  ValuesFunctionFormatCBData(
   void*  cbdata      // pointeur vers l'objet
   )
```

Paramètres

cbdata

[in] Pointeur vers n'importe quel objet pouvant contenir des données supplémentaires sur la conversion des valeurs de l'axe

© 2000-2024, MetaQuotes Ltd.

NameSize (méthode Get)

Retourne la taille de la police de caractères du nom de l'axe.

```
int  NameSize()
```

Valeur de Retour

La taille de la police de caractères du nom de l'axe.

NameSize (méthode Set)

Définit la taille de la police de caractères du nom de l'axe.

```
void  NameSize(
   const int  size      // taille de la police de caractères du nom de l'axe
   )
```

Paramètres

size

[in] Taille de la police de caractères du nom de l'axe.

© 2000-2024, MetaQuotes Ltd.

ZeroLever (méthode Get)

Retourne la valeur du "levier zéro".

```
double  ZeroLever()
```

Valeur de Retour

La valeur du "levier zéro".

Note

La valeur est utilisée pour définir le moment où l'intervalle de l'échelle de l'axe doit être étendu pour inclure la valeur zéro.

ZeroLever (méthode Set)

Définit la valeur du "levier zéro".

```
void  ZeroLever(
   const double  value      // valeur du "levier zéro"
   )
```

Paramètres

value

[in] Valeur du "levier zéro".

Note

La valeur est utilisée pour définir le moment où l'intervalle de l'échelle de l'axe doit être étendu pour inclure la valeur zéro.

© 2000-2024, MetaQuotes Ltd.

DefaultStep (méthode Get)

Retourne la valeur initiale du pas par axe

```
double  DefaultStep()
```

Valeur de Retour

Le pas par axe.

DefaultStep (méthode Set)

Définit la valeur initiale du pas par axe

```
void  DefaultStep(
   const double  value      // pas par axe
   )
```

Paramètres

value

[in] La valeur initiale du pas par axe.

© 2000-2024, MetaQuotes Ltd.

MaxLabels (méthode Get)

Retourne la quantité maximum de nombres affichés sur l'axe.

```
double  MaxLabels()
```

Valeur de Retour

La quantité maximume de nombres sur l'axe.

MaxLabels (méthode Set)

Définit la quantité maximum de nombres affichés sur l'axe.

```
void  MaxLabels(
   const double  value      // quantité maximum
   )
```

Paramètres

value

[in] La quantité maximum de nombres affichés sur l'axe

© 2000-2024, MetaQuotes Ltd.

MinGrace (méthode Get)

Retourne la "tolérance" appliquée au minimum de l'axe.

```
double  MinGrace()
```

Valeur de Retour

La valeur de "tolérance" pour le minimum de l'axe.

Note

Cette valeur est exprimée comme une partie de la longueur totale de l'axe. Par exemple, supposons que les valeurs de l'axe sont situées entre 4.0 et 16.0, sa longueur est donc 12.0. Si MinGrace est égal à 0.1, alors 10% de la longueur de l'axe (ou 1.2) seront déduits de la valeur minimum. L'axe couvre donc en résultat l'intervalle de 2.8 à 16.0.

MinGrace (méthode Set)

Définit la "tolérance" appliquée au minimum de l'axe.

```
void  MinGrace(
   const double  value      // valeur de "tolérance"
   )
```

Paramètres

value

[in] La "tolérance" appliquée au minimum de l'axe.

Note

Cette valeur est exprimée comme une partie de la longueur totale de l'axe. Par exemple, supposons que les valeurs de l'axe sont situées entre 4.0 et 16.0, sa longueur est donc 12.0. Si MinGrace est égal à 0.1, alors 10% de la longueur de l'axe (ou 1.2) seront déduits de la valeur minimum. L'axe couvre donc en résultat l'intervalle de 2.8 à 16.0.

© 2000-2024, MetaQuotes Ltd.

MaxGrace (méthode Get)

Récupère la valeur de "tolérance" appliqué à la valeur maximum de l'axe.

```
double  MaxGrace()
```

Valeur de Retour

La valeur de "tolérance" pour le maximum de l'axe.

Note

Cette valeur est exprimée comme une partie de la longueur totale de l'axe. Par exemple, supposons que les valeurs de l'axe sont situées entre 4.0 et 16.0, sa longueur est donc 12.0. Si MaxGrace est égal à 0.1, alors 10% de la longueur de l'axe (ou 1.2) sont ajoutés à la valeur maximum. L'axe couvre donc en résultat l'intervalle de 4.0 à 17.2.

MaxGrace (méthode Set)

Définit la "tolérance" appliquée à la valveur maximum de l'axe.

```
void  MaxGrace(
   const double  value      // valeur de "tolérance"
   )
```

Paramètres

value

[in] Valeur de "tolérance" appliquée au maximum de l'axe.

Note

Cette valeur est exprimée comme une partie de la longueur totale de l'axe. Par exemple, supposons que les valeurs de l'axe sont situées entre 4.0 et 16.0, sa longueur est donc 12.0. Si MinGrace est égal à 0.1, alors 10% de la longueur de l'axe (ou 1.2) seront déduits de la valeur minimum. L'axe couvre donc en résultat l'intervalle de 2.8 à 16.0.

© 2000-2024, MetaQuotes Ltd.

SelectAxisScale

Met automatiquement à l'échelle l'axe.

```
void  SelectAxisScale()
```

© 2000-2024, MetaQuotes Ltd.

CColorGenerator

La classe CColorGenerator est une classe auxiliaire de la bibliothèque graphique permettant de travailler avec la palette des couleurs.

Description

La classe CColorGenerator contient la palette de couleurs initiale utilisée par défaut pour les courbes (si une couleur n'est pas spécifiée par l'utilisateur).

Si toutes les couleurs de la palette initiale sont déjà utilisées, de nouvelles couleurs sont automatiquement générées et la palette est rechargée.

Déclaration

```
class CColorGenerator
```

Titre

```
#include <Graphics\ColorGenerator.mqh>
```

Méthodes de classe

Méthode	Description
Next	Retourne la couleur suivante de la palette
Reset	Réinitialise le générateur

© 2000-2024, MetaQuotes Ltd.

Next

Retourne la couleur suivante dans la palette.

```
uint  Next()
```

Valeur de Retour

La couleur suivante.

Note

Si toutes les couleurs de la palette ont déjà été passées, de nouvelles couleurs sont automatiquement générées afin de remplacer les anciennes couleurs de la palette.

© 2000-2024, MetaQuotes Ltd.

Reset

Réinitialise le générateur.

```
void  Reset()
```

© 2000-2024, MetaQuotes Ltd.

CCurve

La classe CCurve permet d'utiliser les propriétés des courbes générées sur le graphique.

Description

La classe CCurve définit, installe et récupère les coordonnées et les différentes propriétés des courbes lors de l'utilisation de la classe CGraphic.

Il existe 3 mode de dessin des courbes : points, lignes et histogramme. Des paramètres différents sont implémentés dans la classe pour chaque mode de dessin.

Déclaration

```
class CCurve : public CObject
```

Titre

```
#include <Graphics\Curve.mqh>
```

Hiérarchie d'héritage

CObject
  CCurve

Méthodes de la classe

Méthode	Description
Type	Retourne le type de la courbe
Name	Retourne le nom de la courbe
Color	Retourne la couleur de la courbe
XMax	Retourne la valeur maximum de la fonction X
XMin	Retourne la valeur minimum de la fonction X
YMax	Retourne la valeur maximum de la fonction Y
YMin	Retourne la valeur minimum de la fonction Y
Size	Retourne le nombre de points définissant une courbe
PointsSize	Retourne/définit la taille linéaire des points définissant une courbe
PointsFill	Retourne/définit le flag de remplissage des points définissant une courbe
PointsColor	Retourne/définit la couleur de remplissage des points
GetX	Retourne les valeurs X de tous les points de la courbe dans le tableau

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
GetY	Retourne les valeurs Y de tous les points de la courbe dans le tableau
LinesStyle	Retourne/définit un style de ligne lors du dessin d'une courbe utilisant des lignes
LinesIsSmooth	Retourne/définit le flag de lissage lors du dessin des lignes
LinesSmoothTension	Retourne/définit le paramètre de lissage des courbes lors du dessin utilisant des lignes
LinesSmoothStep	Retourne/définit la longueur des lignes d'approximation pour le lissage lors du dessin utilisant des lignes
LinesWidth	Retourne/définit la largeur d'une ligne lors du dessin d'une courbe utilisant des lignes
HistogramWidth	Retourne/définit la largeur des colonnes lors du dessin utilisant un histogramme
CustomPlotCBData	Retourne/définit un pointeur vers l'objet à utiliser dans le mode de dessin personnalisé des courbes.
CustomPlotFunction	Retourne/définit un pointeur vers la fonction implémentant le mode personnalisé de dessin des courbes.
PointsType	Retourne/définit le flag pointant sur le type de points utilisés lors du dessin d'une courbe sur des points.
StepsDimension	Retourne/définit la valeur indiquant la dimension utilisée dans le rendu de type étape.
TrendLineCoefficients	Retourne/définit les ratios des lignes de tendance pour les écrire dans un tableau.
TrendLineColor	Retourne/définit la couleur d'une ligne de tendance pour une courbe.
TrendLineVisible	Retourne/définit le flag de visibilité d'une ligne de tendance.
Update	Met à jour les coordonnées de la courbe.
Visible	Retourne/définit le flag définissant si une fonction est visible sur le graphique.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Compare

© 2000-2024, MetaQuotes Ltd.

Type

Retourne le type de la courbe.

```
ENUM_CURVE_TYPE  Type()
```

Valeur de Retour

Le type de la courbe.

© 2000-2024, MetaQuotes Ltd.

Name

Retourne le nom de la courbe.

```
string  Name()
```

Valeur de Retour

Le nom de la courbe.

© 2000-2024, MetaQuotes Ltd.

Color

Retourne la couleur de la courbe.

```
uint  Color()
```

Valeur de Retour

La couleur de la courbe.

© 2000-2024, MetaQuotes Ltd.

XMax

Retourne la valeur maximum de la fonction X (nombres réels uniquement).

```
double  XMax()
```

Valeur de Retour

Le nombre réel maximum parmi tous les arguments de la fonction.

© 2000-2024, MetaQuotes Ltd.

XMin

Retourne la valeur minimum de la fonction X (nombres réels uniquement).

```
double  XMin()
```

Valeur de Retour

Le nombre réel minimum parmi tous les arguments de la fonction.

© 2000-2024, MetaQuotes Ltd.

YMax

Retourne la valeur maximum de la fonction Y (nombres réels uniquement).

```
double  YMax()
```

Valeur de Retour

La valeur maximum de la fonction Y (nombres réels uniquement).

© 2000-2024, MetaQuotes Ltd.

YMin

Retourne la valeur minimum de la fonction Y (nombres réels uniquement).

```
double  YMin()
```

Valeur de Retour

La valeur minimum de la fonction Y (nombres réels uniquement).

© 2000-2024, MetaQuotes Ltd.

Size

Retourne le nombre de points définissant la courbe.

```
int  Size()
```

Valeur de Retour

Le nombre de points définissant la courbe.

© 2000-2024, MetaQuotes Ltd.

PointsSize (méthode Get)

Retourne la taille linéaire (en pixels) des points utilisés dans le dessin de la courbe.

```
int  PointsSize()
```

Valeur de Retour

La taille des points en pixels définissant la courbe.

PointsSize (méthode Set)

Définit la taille linéaire (en pixels) des points utilisés dans le dessin de la courbe.

```
void  PointsSize(
   const int  size      // taille des points en pixels
   )
```

Paramètres

size

[in] La taille linéaire (en pixels) des points utilisés pour dessiner la courbe.

© 2000-2024, MetaQuotes Ltd.

PointsFill (méthode Get)

Retourne un flag déterminant si les points de la courbe doivent etre remplis.

```
bool  PointsFill ()
```

Valeur de Retour

La valeur du flag.

Note

true — effectue le remplissage

false — n'effectue pas le remplissage

PointsFill (méthode Set)

Définit un flag déterminant si les points de la courbe doivent etre remplis.

```
void  PointsFill(
   const bool  fill      // valeur du flag
   )
```

Paramètres

fill

[in] Valeur du flag.

Note

true — effectue le remplissage

false — n'effectue pas le remplissage

© 2000-2024, MetaQuotes Ltd.

PointsColor (méthode Get)

Retourne la couleur de remplissage des points.

```
uint  PointsColor ()
```

Valeur de Retour

La couleur de remplissage des points définissant la courbe.

PointsColor (méthode Set)

Définit la couleur de remplissage des points.

```
void  PointsColor(
   const uint  clr      // couleur de remplissage des points
   )
```

Paramètres

clr

[in] La couleur de remplissage des points définissant la courbe.

© 2000-2024, MetaQuotes Ltd.

GetX

Ecrit dans le tableau les coordonnées X de tous les points de la courbe.

```
void  GetX(
   double  &x[]      // Coordonnées X
   )
```

Paramètres

&x[]

[out] Tableau pour l'écriture des coordonnées X.

© 2000-2024, MetaQuotes Ltd.

GetY

Ecrit dans le tableau les coordonnées Y de tous les points de la courbe.

```
void  GetY(
   double  &y[]      // Coordonnées Y
   )
```

Paramètres

&y[]

[out] Tableau pour l'écriture des coordonnées Y.

© 2000-2024, MetaQuotes Ltd.

LinesStyle (méthode Get)

Retourne le style de ligne lors du dessin d'une courbe utilisant des lignes.

```
ENUM_LINE_STYLE  LinesStyle()
```

Valeur de Retour

Le style de ligne.

LinesStyle (méthode Set)

Définit le style de ligne lors du dessin d'une courbe utilisant des lignes.

```
void  LinesStyle (
   ENUM_LINE_STYLE  style      // style de ligne
   )
```

Paramètres

style

[in] Le style de ligne.

© 2000-2024, MetaQuotes Ltd.

LinesIsSmooth (méthode Get)

Retourne un flag définissant si le lissage doit être effectué lors du dessin d'une courbe avec des lignes.

```
bool  LinesIsSmooth()
```

Valeur de Retour

Valeur du flag

Note

true — effectue le lissage

false — n'effectue pas le lissage

LinesIsSmooth (méthode Set)

Définit un flag spécifiant si le lissage doit être effectué lors du dessin d'une courbe avec des lignes.

```
void  LinesIsSmooth(
   const bool  smooth      // valeur du flag
   )
```

Paramètres

smooth

[in] Valeur du flag

Note

true — effectue le lissage

false — n'effectue pas le lissage

© 2000-2024, MetaQuotes Ltd.

LinesSmoothTension (méthode Get)

Retourne le paramètre de lissage de la courbe lors du dessin utilisant des lignes.

```
double  LinesSmoothTension()
```

Valeur de Retour

La valeur du paramètre de lissage

Note

La valeur de 'tension' est dans l'intervalle (0.0; 1.0].

LinesSmoothTension (méthode Set)

Définit le paramètre de lissage de la courbe lors du dessin utilisant des lignes.

```
void  LinesSmoothTension(
   const double  tension      // valeur du paramètre
   )
```

Paramètres

tension

[in] Valeur du paramètre de lissage.

Note

La valeur de 'tension' est dans l'intervalle (0.0; 1.0].

© 2000-2024, MetaQuotes Ltd.

LinesSmoothStep (méthode Get)

Retourne la longueur des lignes d'approximation du lissage lors du dessin par des lignes.

```
double  LinesSmoothStep()
```

Valeur de Retour

La longueur des lignes d'approximation en pixels.

LinesSmoothStep (méthode Set)

Définit la longueur des lignes d'approximation du lissage lors du dessin par des lignes.

```
void  LinesSmoothStep(
   const double  step      // longueur des lignes
   )
```

Paramètres

step

[in] Longueur des lignes d'approximation

© 2000-2024, MetaQuotes Ltd.

LinesWidth (méthode Get)

Retourne la largeur des lignes lors du dessin d'une courbe utilisant des lignes.

```
int  LinesWidth()
```

Valeur de Retour

Largeur des lignes.

LinesWidth (métode Set)

Définit la largeur des lignes lors du dessin d'une courbe utilisant des lignes.

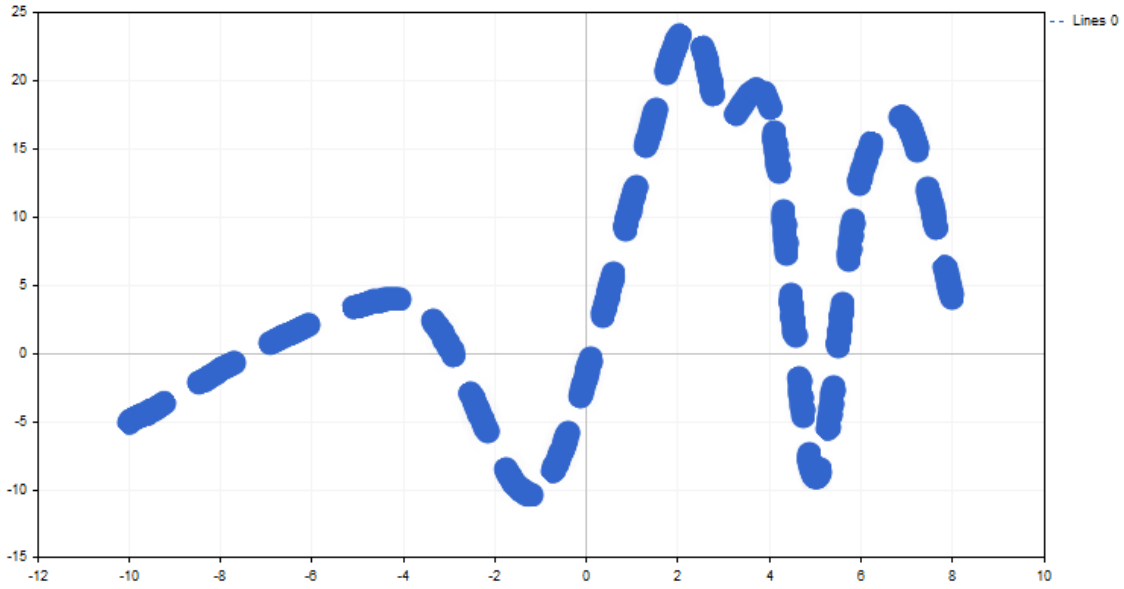
```
void  LinesWidth(
   const int  width      // largeur des lignes
   )
```

Paramètres

width

[in] Largeur des lignes lors du dessin d'une courbe utilisant des lignes.

Exemple :



La largeur d'une ligne a été changée avec le code suivant :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]= { -100,-40,-10,20,30,40,50,60,70,80,120 };
   double y[]= { -5,4,-10,23,17,18,-9,13,17,4,9 };
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"ThickLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"ThickLineGraphic");
     }
//--- crée la courbe
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_LINES);
//--- définit les propriétés de la courbe
   curve.LinesSmooth(true);
   curve.LinesStyle(STYLE_DASH);
   curve.LinesEndStyle(LINE_END_ROUND);
   curve.LinesWidth(10);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

LinesEndStyle (méthode Set)

Retourne le flag indiquant le style de dessin des fins de lignes lors de l'utilisation de lignes pour dessiner une courbe.

```
ENUM_LINE_END  LinesEndStyle()
```

Valeur de Retour

La valeur du flag indiquant le style de dessin des fins de lignes lors de l'utilisation des lignes pour dessiner une courbe.

LinesEndStyle (méthode Get)

Définit le flag indiquant le style de dessin des fins de lignes lors de l'utilisation de lignes pour dessiner une courbe.

```
void  LinesEndStyle(
   ENUM_LINE_END  end_style      // valeur du flag
   )
```

Paramètres

end_style

[in] La valeur du flag indiquant le style de dessin des fins de lignes lors de l'utilisation des lignes pour dessiner une courbe.

© 2000-2024, MetaQuotes Ltd.

HistogramWidth (méthode Get)

Rétourne la largeur des colonnes lors du dessin utilisant un histogramme.

```
int  HistogramWidth()
```

Valeur de Retour

La largeur des colonnes en pixels.

HistogramWidth (méthode Set)

Définit la largeur des colonnes lors du dessin utilisant un histogramme.

```
void  HistogramWidth(
   const int  width      // largeur des colonnes
   )
```

Paramètres

width

[in] Largeur des colonnes en pixels.

© 2000-2024, MetaQuotes Ltd.

CustomPlotCBData (méthode Get)

Retourne un pointeur vers l'objet à utiliser dans le mode de dessin personnalisé des courbes.

```
void*  CustomPlotCBData()
```

Valeur de Retour

Pointeur vers l'objet à utiliser dans le mode de dessin personnalisé des courbes.

CustomPlotCBData (méthode Set)

Définit un pointeur vers l'objet à utiliser dans le mode de dessin personnalisé des courbes.

```
void  CustomPlotCBData(
   void*  cbdata      // pointeur vers l'objet
   )
```

Paramètres

cbdata

[in] Le pointeur vers l'objet à utiliser dans le mode de dessin personnalisé des courbes

© 2000-2024, MetaQuotes Ltd.

CustomPlotFunction (méthode Get)

Retourne un pointeur vers la fonction implémentant le mode personnalisé de dessin des courbes.

```
PlotFucntion  CustomPlotFunction()
```

Valeur de Retour

Pointeur vers la fonction implémentant le mode personnalisé de dessin des courbes.

CustomPlotFunction (méthode Set)

Définit le pointeur vers la fonction implémentant le mode personnalisé de dessin des courbes.

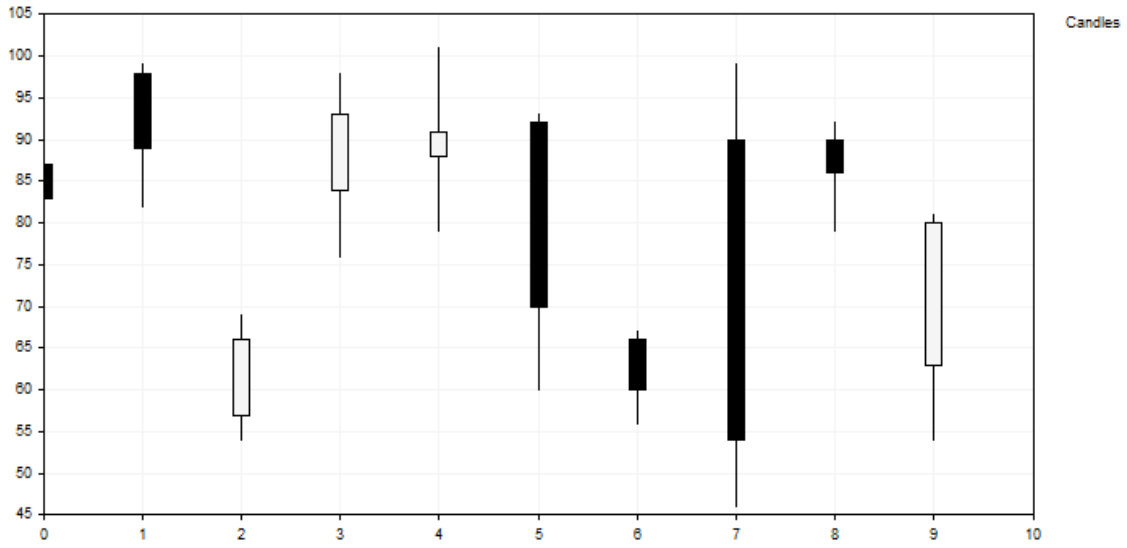
```
void  CustomPlotFunction(
   PlotFucntion  func      // pointeur vers la fonction
   )
```

Paramètres

func

[in] Pointeur vers la fonction implémentant le mode personnalisé de dessin des courbes

Exemple :



Cette courbe constituée de barres est construite avec le mode suivant :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Classe CCandle                                                   |
//| Usage : classe représentant une bougie                           |
//+------------------------------------------------------------------+
class CCandle: public CObject
  {
private:
   double            m_open;
   double            m_close;
   double            m_high;
   double            m_low;
   uint              m_clr_inc;
   uint              m_clr_dec;
   int               m_width;

public:
                     CCandle(const double open,const double close,const double high,co
                                                        const int width,const uint clr_
                    ~CCandle(void);
   double            OpenValue(void)            const { return(m_open);     }
   double            CloseValue(void)           const { return(m_close);    }
   double            HigthValue(void)           const { return(m_high);     }
   double            LowValue(void)             const { return(m_low);      }
   uint              CandleColorIncrement(void) const { return(m_clr_inc);  }
   uint              CandleColorDecrement(void) const { return(m_clr_dec);  }
   int               CandleWidth(void)          const { return(m_width);    }
  };
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CCandle::CCandle(const double open,const double close,const double high,const double l
                                 const int width,const uint clr_inc=0x000000,const uin
                                 m_open(open),m_close(close),m_high(high),m_low(low),
                                 m_clr_inc(clr_inc),m_clr_dec(clr_dec),m_width(width)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CCandle::~CCandle(void)
  {
  }
//+------------------------------------------------------------------+
//| Méthode personnalisé pour dessiner les bougies                   |
//+------------------------------------------------------------------+
void PlotCandles(double &x[],double &y[],int size,CGraphic *graphic,CCanvas *canvas,vo
  {
//--- vérifie l'objet
   CArrayObj *candles=dynamic_cast<CArrayObj*>(cbdata);
   if(candles==NULL || candles.Total()!=size)
      return;
//--- dessine les bougies
   for(int i=0; i<size; i++)
     {
      CCandle *candle=dynamic_cast<CCandle*>(candles.At(i));
```

```
      if(candle==NULL)
         return;
      //--- calcul
      int xc=graphic.ScaleX(x[i]);
      int width_2=candle.CandleWidth()/2;
      int open=graphic.ScaleY(candle.OpenValue());
      int close=graphic.ScaleY(candle.CloseValue());
      int high=graphic.ScaleY(candle.HigthValue());
      int low=graphic.ScaleY(candle.LowValue());
      uint clr=(open<=close) ? candle.CandleColorIncrement() :  candle.CandleColorDec
      //--- dessine la bougie
      canvas.LineVertical(xc,high,low,0x000000);
      //--- dessine le corps réel de la bougie
      canvas.FillRectangle(xc+width_2,open,xc-width_2,close,clr);
      canvas.Rectangle(xc+width_2,open,xc-width_2,close,0x000000);
     }
  }
//+------------------------------------------------------------------+
//| Fonction de lancement du script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   int count=10;
   int width=10;
   double x[];
   double y[];
   ArrayResize(x,count);
   ArrayResize(y,count);
   CArrayObj candles();
   double max=0;
   double min=0;
//--- crée les valeurs
   for(int i=0; i<count; i++)
     {
      x[i] = i;
      y[i] = i;
      //--- calcule les valeurs
      double open=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double close=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double high=MathRound(MathMax(open,close)+(MathRand()/32767.0)*10.0);
      double low=MathRound(MathMin(open,close) -(MathRand()/32767.0)*10.0);
      //--- trouve le max et le min
      if(i==0 || max<high)
         max=high;
      if(i==0 || min>low)
         min=low;
      //--- crée la bougie
      CCandle *candle=new CCandle(open,close,high,low,width);
      candles.Add(candle);
     }
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"CandleGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"CandleGraphic");
     }
//--- crée la courbe
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_CUSTOM,"Candles");
//--- définit les propriétés de la courbe
   curve.CustomPlotFunction(PlotCandles);
   curve.CustomPlotCBData(GetPointer(candles));
```

© 2000-2024, MetaQuotes Ltd.

```
//--- définit les propriétés graphiques
   graphic.YAxis().Max((int)max);
   graphic.YAxis().Min((int)min);
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2024, MetaQuotes Ltd.

PointsType (méthode Get)

Retourne le flag pointant sur le type de points utilisés lors du dessin d'une courbe sur des points.

```
ENUM_POINT_TYPE  PointsType()
```

Valeur de Retour

La valeur du flag indiquant le type de points.

PointsType (méthode Set)

Définit le flag pointant sur le type de points utilisés lors du dessin d'une courbe sur des points.

```
void  PointsType(
   ENUM_POINT_TYPE  type      // valeur du flag
   )
```

Paramètres

type

[in] Une valeur du flag pointant sur le type de points utilisés lors du dessin d'une courbe sur des points.

© 2000-2024, MetaQuotes Ltd.

StepsDimension (méthode Get)

Retourne la valeur indiquant la dimension utilisée dans le rendu de type étape.

```
int  StepsDimension()
```

Valeur de Retour

Dimension utilisée dans le rendu de type étape.

StepsDimension (méthode Set)

Définit la valeur indiquant la dimension utilisée dans le rendu de type étape.

```
void  StepsDimension(
   const int  dimension      // dimension
   )
```

Paramètres

dimension

[in] Dimension (0 ou 1).

Note

0 — x (la ligne horizontale est suivie par la ligne verticale).

1 — y (la ligne verticale est suivie par la ligne horizontale).

© 2000-2024, MetaQuotes Ltd.

TrendLineCoefficients (méthode Get)

Retourne les ratios des lignes de tendance pour les écrire dans un tableau.

```
double&  TrendLineCoefficients()
```

Valeur de Retour

Ratios des lignes de tendance.

TrendLineCoefficients (méthode Set)

Définit les ratios des lignes de tendance pour les écrire dans un tableau.

```
void  TrendLineCoefficients(
   double&  coefficients[]      // tableau pour écrire les ratios
   )
```

Paramètres

coefficients[]

[out] Tableau pour écrire les ratios.

© 2000-2024, MetaQuotes Ltd.

TrendLineColor (méthode Get)

Retourne la couleur d'une ligne de tendance pour une courbe.

```
uint  TrendLineColor()
```

Valeur de Retour

Couleur de la ligne de tendance.

TrendLineColor (méthode Set)

Définit la couleur d'une ligne de tendance pour une courbe.

```
void  TrendLineColor(
   const uint  clr      // couleur de la ligne de tendance
   )
```

Paramètres

clr

[in] Couleur de la ligne

© 2000-2024, MetaQuotes Ltd.

TrendLineVisible (méthode Get)

Retourne le flag de visibilité d'une ligne de tendance.

```
bool  TrendLineVisible()
```

Valeur de Retour

La valeur du flag spéciifant si une ligne de tendance est visible.

TrendLineVisible (méthode Set)

Définit le flag de visibilité d'une ligne de tendance.

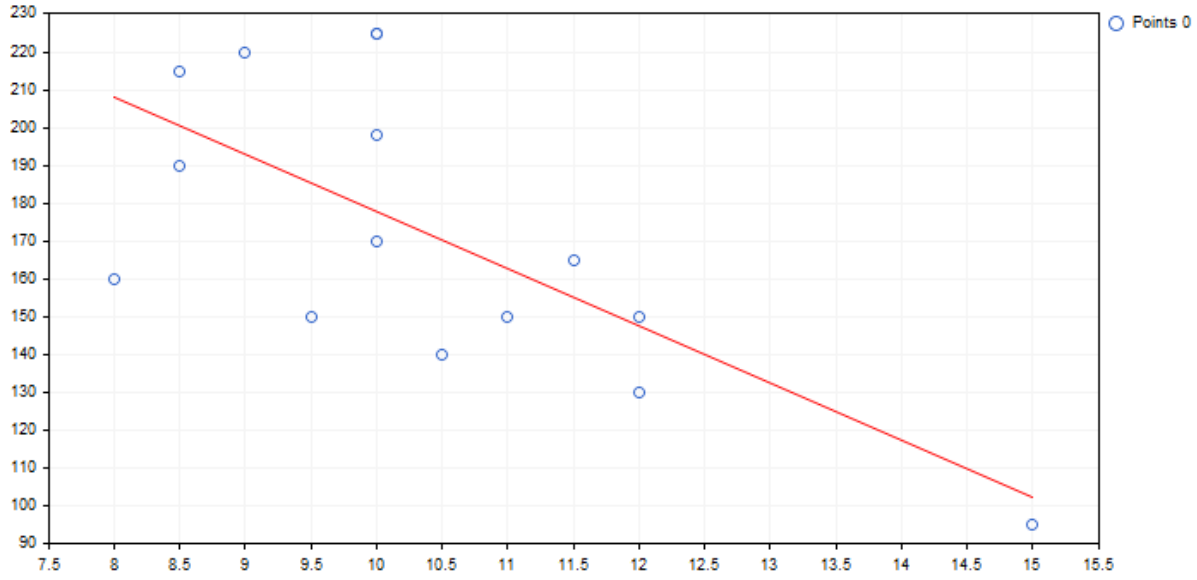
```
void  TrendLineVisible(
   const bool  visible      // valeur du flag
   )
```

Paramètres

visible

[in] Une valeur pour le flag de visibilité de la ligne de tendance.

Exemple :



Le code du dessin de la ligne de tendance sur le graphique se trouve ci-dessous :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              TrendLineGraphic.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Fonction de lancement du programme                               |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]={12.0,11.5,11.0,12.0,10.5,10.0,9.0,8.5,10.0,8.5,10.0,8.0,9.5,10.0,15.0};
   double y[]={130.0,165.0,150.0,150.0,140.0,198.0,220.0,215.0,225.0,190.0,170.0,160.0
//--- crée le graphique
   CGraphic graphic;
   if(!graphic.Create(0,"TrendLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"TrendLineGraphic");
     }
//--- crée la courbe
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_POINTS);
//--- définit les propriétés de la courbe
   curve.TrendLineVisible(true);
   curve.TrendLineColor(ColorToARGB(clrRed));
//--- dessin
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

Update

Met à jour les coordonnées de la courbe.

Version pour travailler avec une coordonnée Y. Les indices du tableau passé sont utilisés comme coordonnées X ici.

```
void  Update(
   const double&  y[]   // Coordonnées Y
   )
```

Cette version utilise les coordonnées X et Y.

```
void  Update(
   const double&  x[],     // Coordonnées X
   const double&  y[]      // Coordonnées Y
   )
```

Version pour travailler avec des points CPoint2D.

```
void  Update(
   const CPoint2D&  points[]      // Coordonnées de la courbe
   )
```

Version pour travailler avec un pointeur vers la fonction CurveFunction.

```
void  Update(
   CurveFunction  function,     // pointeur vers la fonction décrivant une courbe
   const double   from,         // valeur initiale de l'argument de la fonction
   const double   to,           // valeur finale de l'argument de la fonction
   const double   step          // incrément de l'argument
   )
```

Paramètres

x[]

[in] Coordonnées X.

y[]

[in] Coordonnées Y.

points[]

[in] Coordonnées de la courbe.

function

[in] Un pointeur vers la fonction décrivant un courbe

from

[in] Valeur initiale de l'argument de la fonction

to

[in] Valeur de fin de l'argument de la fonction

step

© 2000-2024, MetaQuotes Ltd.

[in] Incrément de l'argument

© 2000-2024, MetaQuotes Ltd.

Visible (méthode Get)

Retourne le flag définissant si une fonction est visible sur le graphique.

```
void  Visible(
   const bool  visible      //
   )
```

Valeur de Retour

Une valeur du flag définissant la visibilité d'une fonction sur le graphique.

Visible (méthode Set)

Définit le flag définissant si une fonction est visible sur le graphique.

```
void  Visible(
   const bool  visible      // valeur du flag
   )
```

Paramètres

visible

[in] Une valeur du flag définissant la visibilité d'une fonction sur le graphique.

© 2000-2024, MetaQuotes Ltd.

CGraphic

CGraphic est une classe de base pour créer des graphiques personnalisés.

Description

La classe CGraphic fournit de nombreux éléments permettant de travailler avec les graphiques personnalisés.

La classe stocke les éléments principaux du graphique, définit leurs paramètres et les dessine.

La classe stocke également les courbes du graphique et fournit différentes options d'affichage.

Déclaration

```
class CGraphic
```

Titre

```
#include <Graphics\Graphic.mqh>
```

Méthodes de la classe

Méthode	Description
Create	Crée une ressource graphique liée à un objet Graphique
Destroy	Supprime un graphique et détruit une ressource graphique
Update	Affiche les nouveaux changements uniquement
ChartObjectName	Retourne le nom d'un objet lié à un graphique
ResourceName	Retourne le nom de la ressource graphique
XAxis	Retourne le pointeur vers l'axe X
YAxis	Retourne le pointeur vers l'axe Y
GapSize	Retourne/définit la taille des indentations entre les éléments du graphique
BackgroundColor	Retourne/définit la couleur d'arrière plan
BackgroundMain	Retourne/définit l'en-tête du graphique
BackgroundMainSize	Retourne/définit la taille de la police de caractères de l'en-tête
BackgroundMainColor	Retourne/définit la couleur de l'en-tête du graphique
BackgroundSub	Retourne/définit un sous en-tête

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
BackgroundSubSize	Retourne/définit la taille de la police de caractères d'un sous en-tête
BackgroundSubColor	Retourne/définit la couleur d'un sous en-tête du graphique
GridLineColor	Retourne/définit la couleur des lignes de la grille
GridBackgroundColor	Retourne/définit la couleur d'arrière plan de la grille
GridCircleRadius	Retourne/définit le rayon des points des noeuds de la grille
GridCircleColor	Retourne/définit la couleur des points des noeuds de la grille
GridHasCircle	Retourne/définit le flag d'affichage des points des noeuds de la grille
GridAxisLineColor	Retourne la valeur de la couleur des axes réels du graphique.
HistoryNameWidth	Retourne/définit la longueur maximum autorisée pour afficher le nom d'une courbe
HistoryNameSize	Retourne/définit la taille de la police de caractères du nom d'une courbe
HistorySymbolSize	Retourne/définit la taille des symboles de conventions de notation
TextAdd	Ajoute un texte au graphique
LineAdd	Ajoute une ligne au graphique
CurveAdd	Crée et ajoute une courbe sur le graphique
CurvePlot	Dessine une courbe créée auparavant et référencée par son indice
CurvePlotAll	Dessine toutes les courbes créées auparavant
CurveGetByIndex	Retourne une courbe par son indice spécifié
CurveGetByName	Retourne une courbe par son nom spécifié
CurveRemoveByIndex	Supprime une courbe par son indice spécifié.
CurveRemoveByName	Supprime une courbe par son nom spécifié.
CurvesTotal	Retourne le nombre de courbes pour le graphique donné.
MarksToAxisAdd	Ajoute une marque d'échelle sur l'axe du graphique
MajorMarkSize	Retourne/définit la taille des ticks de l'échelle sur les axes du graphique

© 2000-2024, MetaQuotes Ltd.

Méthode	Description
FontSet	Définit les paramètres de la police de caractères actuelle
FontGet	Retourne les paramètres de la police de caractères actuelle
Attach	Retourne/définit une ressource graphique et la lie à l'instance de classe CGraphic
CalculateMaxMinValues	Calcule (re-calcule) les valeurs minimum et maximum du graphique sur les deux axes.
Height	Retourne la hauteur du graphique en pixels.
IndentDown	Retourne/définit l'indentation du graphique depuis la bordure inférieure.
IndentLeft	Retourne/définit l'indentation du graphique depuis la bordure gauche.
IndentRight	Retourne/définit l'indentation du graphique depuis la bordure droite.
IndentUp	Retourne/définit l'indentation du graphique depuis la bordure supérieure.
Redraw	Redessine le graphique.
ResetParameters	Réinitialisae les paramètres de dessin du graphique.
ScaleX	Met les valeurs de l'axe X à l'échelle.
ScaleY	Met les valeurs de l'axe Y à l'échelle.
SetDefaultParameters	Définit les paramètres du graphique aux valeurs par défaut.
Width	Retourne la largeur du graphique en pixels.

© 2000-2024, MetaQuotes Ltd.

Create

Crée une ressource graphique liée à un objet graphique.

```
bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nome
   const int     subwin,      // indice de la sous-fenêtre
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y1
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom.

subwin

[in] Indice de la sous-fenêtre.

x1

[in] Coordonnée X1.

y1

[in] Coordonnée Y1.

x2

[in] Coordonnée X2.

y2

[in] Coordonnée Y2.

© 2000-2024, MetaQuotes Ltd.

Destroy

Supprime un graphique et détruit une ressource graphique.

```
void  Destroy()
```

© 2000-2024, MetaQuotes Ltd.

Update

Affiche les changements implémentés.

```
void  Update(
   const bool  redraw=true      // flag
   )
```

Paramètres

redraw=true

[in] Valeur du flag

© 2000-2024, MetaQuotes Ltd.

ChartObjectName

Retourne le nom d'un objet lié à un graphique.

```
string  ChartObjectName()
```

Valeur de Retour

Le nom d'un objet lié à un graphique.

© 2000-2024, MetaQuotes Ltd.

ResourceName

Récupère le nom d'une ressource graphique.

```
string  ResourceName()
```

Valeur de Retour

Le nom d'une ressource graphique.

© 2000-2024, MetaQuotes Ltd.

XAxis

Retourne un pointeur vers l'axe X.

```
CAxis  *XAxis()
```

Valeur de Retour

Pointeur sur l'axe X.

© 2000-2024, MetaQuotes Ltd.

YAxis

Retourne un pointeur vers l'axe Y.

```
CAxis  *YAxis()
```

Valeur de Retour

Pointeur vers l'axe Y.

© 2000-2024, MetaQuotes Ltd.

GapSize (méthode Get)

Retourne la taille des tirets entre les éléments du graphique.

```
int  GapSize()
```

Valeur de Retour

Taille des tirets en pixels.

GapSize (méthode Set)

Définit la taille des tirets entre les éléments du graphique.

```
void  GapSize(
   const int  size     // taille des tirets
   )
```

Paramètres

size

[in] Taille des tirets en pixels.

© 2000-2024, MetaQuotes Ltd.

BackgroundColor (méthode Get)

Retourne la couleur d'arrière plan.

```
color  BackgroundColor()
```

BackgroundColor (méthode Set)

Définit la couleur d'arrière plan.

```
void  BackgroundColor(
   const color  clr      // couleur d'arrière plan
   )
```

Paramètres

clr

[in] Couleur d'arrière plan.

© 2000-2024, MetaQuotes Ltd.

BackgroundMain (méthode Get)

Retourne l'en-tête d'un graphique

```
string  BackgroundMain()
```

BackgroundMain (méthode Set)

Définit le texte d'un header du graphique.

```
void  BackgroundMain(
   const string  main      // texte de l'en-tête
   )
```

Paramètres

main

[in] Texte de l'en-tête d'un graphique

© 2000-2024, MetaQuotes Ltd.

BackgroundMainSize (méthode Get)

Retourne la taille de la police de caractères de l'en-tête.

```
int  BackgroundMainSize()
```

Valeur de Retour

La taille de la police de caractères de l'en-tête.

BackgroundMainSize (méthode Set)

Définit la taille de la police de caractères de l'en-tête.

```
void  BackgroundMainSize(
   const int  size     // taille de l'en-tête
   )
```

Paramètres

size

[in] La taille de la police de caractères de l'en-tête.

© 2000-2024, MetaQuotes Ltd.

BackgroundMainColor (méthode Get)

Retourne la couleur de l'en-tête.

```
color  BackgroundMainColor()
```

Valeur de Retour

Couleur de l'en-tête.

BackgroundMainColor (méthode Set)

Définit la couleur de l'en-tête.

```
void  BackgroundMainColor(
   const color  clr      // couleur de l'en-tête
   )
```

Paramètres

clr

[in] Couleur de l'en-tête.

© 2000-2024, MetaQuotes Ltd.

BackgroundSub (méthode Get)

Retourne le texte du sous en-tête.

```
string  BackgroundSub()
```

Valeur de Retour

Le texte du sous en-tête.

BackgroundSub (méthode Set)

Définit le texte du sous en-tête.

```
void  BackgroundSub (string sub)(
   const string  sub      // texte du sous en-tête
   )
```

Paramètres

sub

[in] Le texte du sous en-tête.

© 2000-2024, MetaQuotes Ltd.

BackgroundSubSize (méthode Get)

Retourne la taille de la police de caractères du sous en-tête.

```
int BackgroundSubSize()
```

BackgroundSubSize (méthode Get)

Définit la taille de la police de caractères du sous en-tête.

```
void  BackgroundSubSize(
   const int  size      // taille de la police de caractères du sous en-tête
   )
```

Paramètres

size

[in] Taille de la police de caractères du sous en-tête

© 2000-2024, MetaQuotes Ltd.

BackgroundSubColor (méthode Get)

Retourne la couleur du sous en-tête.

```
color  BackgroundSubColor()
```

BackgroundSubColor (méthode Set)

Définit la couleur du sous en-tête.

```
void  BackgroundSubColor(
   const color  clr      // couleur du sous en-tête
   )
```

Paramètres

clr

[in] Couleur du sous en-tête.

© 2000-2024, MetaQuotes Ltd.

GridLineColor (méthode Get)

Retourne la couleur des lignes de la grille

```
color  GridLineColor()
```

Valeur de Retour

Couleur des lignes de la grille.

GridLineColor (méthode Set)

Définit la couleur des lignes de la grille.

```
void  GridLineColor(
   const color  clr      // couleur des lignes
   )
```

Paramètres

clr

[in] Couleur des lignes de la grille.

© 2000-2024, MetaQuotes Ltd.

GridBackgroundColor (méthode Get)

Retourne la couleur d'arrière plan de la grille.

```
color  GridBackgroundColor()
```

Valeur de Retour

La couleur d'arrière plan de la grille

GridBackgroundColor (méthode Set)

Définit la couleur d'arrière plan de la grille

```
void  GridBackgroundColor(
   const color  clr      // couleur d'arrière plan de la grille
   )
```

Paramètres

clr

[in] Couleur d'arrière plan de la grille

© 2000-2024, MetaQuotes Ltd.

GridCircleRadius (méthode Get)

Retourne le rayon du des points des noeuds de la grille.

```
int  GridCircleRadius()
```

Valeur de Retour

Rayon des points en pixels.

GridCircleRadius (méthode Set)

Définit le rayon des points des noeuds de la grille

```
void  GridCircleRadius(
   const int  r      // rayon
   )
```

Paramètres

r

[in] Rayon des points en pixels.

© 2000-2024, MetaQuotes Ltd.

GridCircleColor (méthode Get)

Retourne la couleur des points des noeuds de la grille.

```
color  GridCircleColor()
```

Valeur de Retour

La couleur des points.

GridCircleColor (méthode Set)

Définit la couleur des points des noeuds de la grille.

```
void  GridCircleColor(
   const color  clr      // couleur des points
   )
```

Paramètres

clr

[in] La couleur des points.

© 2000-2024, MetaQuotes Ltd.

GridHasCircle (méthode Get)

Retourne le flag définissant si les points des noeuds de la grille doivent être affichés.

```
bool  GridHasCircle()
```

Valeur de Retour

La valeur du flag.

Note

true — affiche les points

false — n'affiche pas les points

GridHasCircle (méthode Set)

Définit le flag définissant si les points des noeuds de la grille doivent être affichés.

```
void  GridHasCircle(
   const bool  has
   )
```

Paramètres

has

[in] Valeur du flag.

Note

true — affiche les points

false — n'affiche pas les points

© 2000-2024, MetaQuotes Ltd.

GridAxisLineColor (méthode Get)

Retourne la valeur de la couleur des axes réels du graphique.

```
uint  GridAxisLineColor()
```

Valeur de Retour

Couleur des axes réels du graphique.

GridAxisLineColor (méthode Set)

Définit la valeur de la couleur des axes réels du graphique.

```
void  GridAxisLineColor(
   const uint  clr      // couleur des axes du graphique
   )
```

Paramètres

clr

[in] Couleur des axes réels du graphique.

© 2000-2024, MetaQuotes Ltd.

HistoryNameWidth (méthode Get)

Retourne la longueur maximum autorisée pour l'affichage du nom d'une courbe.

```
int  HistoryNameWidth()
```

Valeur de Retour

Longueur maximum en pixels.

Note

Si le nom de la courbe est supérieur à la longueur maximum autorisée, il est tronqué et des points sont ajoutés à la fin.

HistoryNameWidth (méthode Set)

Définit la longueur maximum autorisée pour l'affichage du nom d'une courbe.

```
void  HistoryNameWidth(
   const int  width      // longueur maximum
   )
```

Paramètres

width

[in] Longueur maximum en pixels.

Note

Si le nom de la courbe est supérieur à la longueur maximum autorisée, il est tronqué et des points sont ajoutés à la fin.

© 2000-2024, MetaQuotes Ltd.

HistoryNameSize (méthode Get)

Retourne la taille de la police de caractères du nom de la courbe.

```
int  HistoryNameSize()
```

Valeur de Retour

La taille de la police de caractères du nom de la courbe.

HistoryNameSize (méthode Set)

Définit la taille de la police de caractères du nom de la courbe.

```
void  HistoryNameSize (int size)(
   const int  size      // taille de la police de caractères du nom
   )
```

Paramètres

size

[in] Taille de la police de caractères du nom.

© 2000-2024, MetaQuotes Ltd.

HistorySymbolSize (méthode Get)

Retourne la taille des symboles de convention de notation

```
int  HistorySymbolSize()
```

Valeur de Retour

La taille des symboles de convention de notation

HistorySymbolSize (méthode Set)

Définit la taille des symboles de convention de notation

```
void  HistorySymbolSize(
   const int  size      // taille des symboles
   )
```

Paramètres

size

[in] Taille des symboles de convention de notation.

© 2000-2024, MetaQuotes Ltd.

TextAdd

Ajout d'un texte sur un graphique.

Version utilisant les coordonnées X et Y

```
void  TextAdd(
   const int     x,              // coordonnée X
   const int     y,              // coordonnée Y
   const string  text,           // texte
   const uint    clr,            // couleur
   const uint    alignment=0     // alignement
   )
```

Version avec CPoint

```
void  TextAdd(
   const CPoint  &point,         // coordonnée du point
   const string  text,           // texte
   const uint    clr,            // couleur
   const uint    alignment=0     // alignement
   )
```

Paramètres

x

[in] Coordonnée X.

y

[in] Coordonnée Y.

&point

[in] Coordonnées du point.

text

[in] Texte.

clr

[in] Couleur.

alignment=0

[in] Alignement.

© 2000-2024, MetaQuotes Ltd.

LineAdd

Ajoute une ligne dans un graphique.

Cette version utilise les coordonnées X et Y

```
void  LineAdd(
   const int   x1,         // coordonnée x1
   const int   y1,         // coordonnée y1
   const int   x2,         // coordonnée x2
   const int   y2,         // coordonnée y2
   const uint  clr,        // couleur
   const uint  style       // style
   )
```

Version avec CPoint

```
void  LineAdd2(
   const CPoint  &point1,      // coordoonées du premier point
   const CPoint  &point2,      // coordonnées du second point
   const uint    clr,          // couleur
   const uint    style         // style
   )
```

Paramètres

x1

[in] Coordonnée X1.

y1

[in] Coordonnée Y1.

x2

[in] Coordonnée X2.

y2

[in] Coordonnée Y2.

&point1

[in] Coordonnées du premier point.

&point2

[in] Coordonnées du second point.

clr

[in] Couleur.

style

[in] Style.

© 2000-2024, MetaQuotes Ltd.

CurveAdd

Crée et ajoute une nouvelle courbe au graphique.

Cette version utilise la coordonnée Y (la couleur de la courbe est définie automatiquement)

```
CCurve* CurveAdd(
   const double      &y[],          // coordonnée Y
   ENUM_CURVE_TYPE   type,          // type de la courbe
   const string      name=NULL      // nom de la courbe
   )
```

Note

Les indices du tableau y sont utilisés comme coordonnées X pour la courbe.

Cette version utilise les coordonnées X et Y (la couleur de la courbe est définie automatiquement)

```
CCurve*  CurveAdd(
   const double      &x[],          // coordonnées X
   const double      &y[],          // coordonnée Y
   ENUM_CURVE_TYPE   type,          // type de la courbe
   const string      name=NULL      // nom de la courbe
   )
```

La version pour travailler avec des points CPoint2D (la couleur de la courbe est définie automatiquement)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // coordonnées des points
   ENUM_CURVE_TYPE   type,          // type de la courbe
   const string      name=NULL      // nom de la courbe
   )
```

Version utilisant un pointeur vers la fonction CurveFunction (la couleur de la courbe est définie automatiquement)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // pointeur vers la fonction
   const double      from,          // valeur initiale de l'argument
   const double      to,            // valeur de fin de l'argument
   const double      step,          // incrément de l'argument
   ENUM_CURVE_TYPE   type,          // type de la courbe
   const string      name=NULL      // nom de la courbe
   )
```

© 2000-2024, MetaQuotes Ltd.

Version utilisant la coordonnée Y (la couleur de la courbe est définie par l'utilisateur)

```
CCurve*  CurveAdd(
   const double      &y[],          // coordonnée Y
   const uint        clr,           // couleur de la courbe
   ENUM_CURVE_TYPE   type,          // type de la courbe
   const string      name=NULL      // nom de la courbe
   )
```

Note

Les indices du tableau y sont utilisés comme coordonnées X pour la courbe.

Cette utilise les coordonnées X et Y (la couleur de la courbe est définie par l'utilisateur)

```
CCurve*  CurveAdd(
   const double      &x[],          // coordonnées X
   const double      &y[],          // coordonnée Y
   const uint        clr,           // couleur de la courbe
   ENUM_CURVE_TYPE   type,          // type de la courbe
   const string      name=NULL      // nom de la courbe
   )
```

Version utilisant des points CPoint2D (la couleur de la courbe est définie par l'utilisateur)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // coordonnées des points
   const uint        clr,           // couleur de la courbe
   ENUM_CURVE_TYPE   type,          // type de la courbe
   const string      name=NULL      // nom de la courbe
   )
```

Version utilisant un pointeur sur la fonction CurveFunction (la couleur de la courbe est définie par l'utilisateur)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // pointeur vers la fonction
   const double      from,          // valeur initiale de l'argument
   const double      to,            // valeur de fin de l'argument
   const double      step,          // incrément de l'argument
   const uint        clr,           // couleur de la courbe
   ENUM_CURVE_TYPE   type,          // type de la courbe
   const string      name=NULL      // nom de la courbe
   )
```

Paramètres

© 2000-2024, MetaQuotes Ltd.

&x[]

[in] Coordonnée X.

&y[]

[in] Coordonnée Y.

&points[]

[in] Coordonnées des points.

function

[in] Pointeur vers la fonction.

from

[in] Valeur initiale de l'argument.

to

[in] Valeur finale de l'argument.

step

[in] Incrément de l'argument.

type

[in] Type de courbe.

name=NULL

[in] Nom de la courbe.

clr

[in] Couleur de la courbe.

Valeur de Retour

Un pointeur vers la courbe créée.

© 2000-2024, MetaQuotes Ltd.

CurvePlot

Affiche la courbe avec l'indice spécifié créé précédemment.

```
bool  CurvePlot(
   const int  index      // indice
   )
```

Paramètres

index

[in] Indice de la courbe

Valeur de Retour

true - en cas de succès, sinon - false.

© 2000-2024, MetaQuotes Ltd.

CurvePlotAll

Affiche toutes les courbes ajoutées précédemment à un graphique.

```
bool  CurvePlotAll()
```

Valeur de Retour

true - en cas de succès, sinon - false.

© 2000-2024, MetaQuotes Ltd.

CurveGetByIndex

Retourne la courbe par un indice spécifié.

```
CCurve*  CurveGetByIndex(
   const int  index      // indice de la courbe
   )
```

Paramètres

index

[in] Indice de la courbe.

Valeur de Retour

Pointeur vers la courbe avec l'indice spécifié.

© 2000-2024, MetaQuotes Ltd.

CurveGetByName

Retourne une courbe par son nom spécifié.

```
CCurve*  CurveGetByName(
   const string  name      // nom de la courbe
   )
```

Paramètres

name

[in] Nom de la courbe.

Valeur de Retour

Pointeur vers la première courbe trouvée avec le nom spécifié.

© 2000-2024, MetaQuotes Ltd.

CurvesTotal

Retourne le nombre de courbes pour le graphique donné.

```
int  CurvesTotal()
```

Valeur de Retour

Nombre de courbes.

Note

Toutes les courbes du graphique courant sont prises en compte, indépendamment qu'elles soient dessinées et/ou visible.

© 2000-2024, MetaQuotes Ltd.

CurveRemoveByName

Supprime une courbe par son nom spécifié.

```
bool  CurveRemoveByName(
   const string  name      // nom de la courbe
   )
```

Paramètres

name

[in] Nom de la courbe à supprimer.

Valeur de Retour

true — en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

CurveRemoveByIndex

Supprime une courbe par son indice spécifié.

```
bool  CurveRemoveByIndex(
   const int  index      // indice de la courbe
   )
```

Paramètres

index

[in] Indice de la courbe à supprimer.

Valeur de Retour

true — en cas de succès, sinon — false.

© 2000-2024, MetaQuotes Ltd.

MarksToAxisAdd

Ajoute une marque d'échelle (ticks) sur l'axe spécifié du graphique.

```
bool  MarksToAxisAdd(
   const double        &marks[],        // coordonnées du tick
   const int           mark_size,       // taille du tick
   ENUM_MARK_POSITION  position,        // position du tick
   const int           dimension=0      // dimension
   )
```

Paramètres

&marks[]

[in] Coordonnées du tick

mark_size

[in] Taille du tick

position

[in] Position du tick

dimension=0

[in] 0 — ajout sur l'axe X,

1 — ajout sur l'axe Y

Valeur de Retour

true - en cas de succès, sinon - false.

© 2000-2024, MetaQuotes Ltd.

MajorMarkSize (méthode Get)

Retourne la taille des ticks de l'échelle sur l'axe des coordonnées.

```
int  MajorMarkSize()
```

MajorMarkSize (méthode Set)

Définit la taille des ticks de l'échelle sur l'axe des coordonnées.

```
void  MajorMarkSize(
   const int  size      // taille des ticks
   )
```

Paramètres

size

[in] Taille des ticks en pixels.

© 2000-2024, MetaQuotes Ltd.

FontSet

Définit les paramètres de la police de caractères actuelle.

```
bool  FontSet(
   const string  name,         // nom
   const int     size,         // taille
   const uint    flags=0,      // flags
   const uint    angle=0       // angle
   )
```

Paramètres

name

[in] Nom.

size

[in] Taille.

flags=0

[in] Flags.

angle=0

[in] Angle.

Valeur de Retour

true - en cas de succès, sinon - false.

© 2000-2024, MetaQuotes Ltd.

FontGet

Retourne les paramètres de la police de caractères actuelle.

```
void  FontGet(
   string  &name,       // nom
   int     &size,       // taille
   uint    &flags,      // flags
   uint    &angle       // angle
   )
```

Paramètres

&name

[out] Nom.

&size

[out] Taille.

&flags

[out] Flags.

&angle

[out] Angle.

© 2000-2024, MetaQuotes Ltd.

Attach

Version retournant une ressource graphique depuis l'objet OBJ_BITMAP_LABEL et le rattachant à l'instance de classe CGraphic :

```
bool  Attach(
   const long    chart_id,     // identifiant du graphique
   const string  objname       // nom de l'objet graphique
   )
```

Version créant une ressource graphique pour l'obje OBJ_BITMAP_LABEL et le rattachant à l'instance de classe CGraphic :

```
bool  Attach(
   const long    chart_id,     // identifiant du graphique
   const string  objname,      // nom de l'objet graphique
   const int     width,        // largeur de l'image
   const int     height        // hauteur de l'image
   )
```

Paramètres

chart_id

[in] Identifiant du graphique.

objname

[in] Nom de l'objet graphique.

width

[in] Largeur de l'image dans la ressource.

height

[in] Hauteur de l'image dans la ressource.

Valeur de Retour

true — en cas de succès, false — en cas d'échec du rattachement de l'objet.

© 2000-2024, MetaQuotes Ltd.

CalculateMaxMinValues

Calcule (re-calcule) les valeurs minimum et maximum du graphique sur les deux axes.

```
void  CalculateMaxMinValues()
```

© 2000-2024, MetaQuotes Ltd.

Height

Retourne la hauteur du graphique en pixels.

```
int  Height()
```

Valeur de Retour

Hauteur du graphique en pixels.

© 2000-2024, MetaQuotes Ltd.

IndentDown (méthode Get)

Retourne l'indentation du graphique depuis la bordure inférieure.

```
int  IndentDown()
```

Valeur de Retour

Taille de l'indentation en pixels.

IndentDown (méthode Set)

Définit l'indentation du graphique depuis la bordure inférieure.

```
void  IndentDown(
   const int  down      // taille de l'indentation
   )
```

Paramètres

down

[in] Taille de l'indentation en pixels.

© 2000-2024, MetaQuotes Ltd.

IndentLeft (méthode Get)

Retourne l'indentation du graphique depuis la bordure gauche.

```
int  IndentLeft()
```

Valeur de Retour

Taille de l'indentation en pixels.

IndentLeft (méthode Set)

Définit l'indentation du graphique depuis la bordure gauche.

```
void  IndentLeft(
   const int  left      // taille de l'indentation
   )
```

Paramètres

left

[in] Taille de l'indentation en pixels.

© 2000-2024, MetaQuotes Ltd.

IndentRight (méthode Get)

Retourne l'indentation du graphique depuis la bordure droite.

```
int  IndentRight()
```

Valeur de Retour

Taille de l'indentation en pixels.

IndentRight (méthode Set)

Définit l'indentation du graphique depuis la bordure droite.

```
void  IndentRight(
   const int  right      // taille de l'indentation
   )
```

Paramètres

right

[in] Taille de l'indentation en pixels.

© 2000-2024, MetaQuotes Ltd.

IndentUp (méthode Get)

Retourne l'indentation du graphique depuis la bordure supérieure.

```
int  IndentUp()
```

Valeur de Retour

Taille de l'indentation en pixels.

IndentUp (méthode Set)

Définit l'indentation du graphique depuis la bordure supérieure.

```
void  IndentUp(
   const int  up      // taille de l'indentation
   )
```

Paramètres

up

[in] Taille de l'indentation en pixels.

© 2000-2024, MetaQuotes Ltd.

Redraw

Redessine le graphique.

```
bool  Redraw(
   const bool  rescale=false      // valeur du flag
   )
```

Paramètres

rescale=false

[in] Le flag indiquant si un graphique doit être remis à l'échelle.

Valeur de Retour

true - en cas de succès, sinon - false.

© 2000-2024, MetaQuotes Ltd.

ResetParameters

Réinitialisae les paramètres de dessin du graphique.

```
void  ResetParameters()
```

© 2000-2024, MetaQuotes Ltd.

ScaleX

Met les valeurs de l'axe X à l'échelle.

```
virtual int  ScaleX(
   double  x      // valeur pour l'axe X
   )
```

Paramètres

x

[in] Valeur réelle pour l'axe X.

Valeur de Retour

Une valeur en pixels.

Note

Met à l'échelle une valeur réelle en pixel pour un affichage sur le graphique.

© 2000-2024, MetaQuotes Ltd.

ScaleY

Met les valeurs de l'axe Y à l'échelle.

```
virtual int  ScaleY(
   double  y      // valeur pour l'axe Y
   )
```

Paramètres

y

[in] Valeur réelle pour l'axe Y.

Valeur de Retour

Une valeur en pixels.

Note

Met à l'échelle une valeur réelle en pixel pour un affichage sur le graphique.

© 2000-2024, MetaQuotes Ltd.

SetDefaultParameters

Définit les paramètres du graphique aux valeurs par défaut.

```
void  SetDefaultParameters()
```

© 2000-2024, MetaQuotes Ltd.

Width

Retourne la largeur du graphique en pixels.

```
int  Width()
```

Valeur de Retour

Largeur du graphique en pixels.

© 2000-2024, MetaQuotes Ltd.

Indicateurs Techniques et Séries de Données

Cette section contient les détails techniques d'utilisation des indicateurs techniques et des séries de données, ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation des classes des indicateurs techniques et des séries de données classes vous fera gagner du temps lors du dévelopement d'applications (scripts, Experts Advisor).

La Bibliothèque Standard MQL5 est située dans le répertoire de travail du terminal dans le répertoire Include\Indicators.

Classe/groupe	Description
Classes de base	Groupe de base et classes annexes
Classes des séries de données	Groupe de classes des séries de données
Indicateurs de Tendance	Groupe de classes d'indicateurs de Tendance
Oscillateurs	Groupe de classes d'Oscillateurs
Indicateurs de Volumes	Groupe de classes d'indicateurs de Volume
Indicateurs de Bill Williams	Classes d'indicateurs de Bill Williams
Indicateurs personnalisés	Classe d'indicateurs personnalisés

© 2000-2024, MetaQuotes Ltd.

Classes de Base et Classes Annexes des Indicateurs Techniques et des Séries de Données

Cette section contient les détails techniques des classes de base et des classes annexes des indicateurs techniques et des séries de données, ainsi qu'une description des composants correspondant de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Classe/groupe	Description
CSpreadBuffer	Classe du buffer de l'historique du spread
CTimeBuffer	Classe du buffer de l'historique des prix d'ouverture
CTickVolumeBuffer	Classe du buffer de l'historique des volumes des ticks
CRealVolumeBuffer	Classe du buffer de l'historique des volumes réels
CDoubleBuffer	Classe de base du buffer de données de type double
COpenBuffer	Classe du buffer des prix d'ouverture
CHighBuffer	Classe du buffer des prix les plus hauts
CLowBuffer	Classe du buffer des prix les plus bas
CCloseBuffer	Classe du buffer des prix de clôture
CIndicatorBuffer	Classe du buffer de l'indicateur technique
CSeries	Classe de base pour l'accès aux données des séries de données
CPriceSeries	Classe de base pour l'accès aux prix
CIndicator	Classe de base d'un indicateur technique
CIndicators	Collection d'indicateurs techniques et de séries de données

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

CSpreadBuffer

CSpreadBuffer est une classe permettant un accès simplifié aux spreads des barres de l'historique.

Description

La classe CSpreadBuffer fournit l'accès aux spreads des barres de l'historique.

Déclaration

```
class CSpreadBuffer: public CArrayInt
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayInt
      CSpreadBuffer

Méthodes de Classe

Attributs	
Size	Définit la taille du buffer
Paramètres	
SetSymbolPeriod	Définit le symbole et la période
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer par son index.
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayInt

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2024, MetaQuotes Ltd.

Size

Définit la taille du buffer.

```
void  Size(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Définit le symbole et la période.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbole
   const ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Nouveau symbole.

period

[in] Nouvelle période (énumération ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du buffer par son index.

```
int  At(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

CTimeBuffer

CTimeBuffer est une classe permettant un accès simplifié aux prix d'ouverture des barres de l'historique.

Description

La classe CTimeBuffer fournit l'accès aux prix d'ouverture des barres de l'historique.

Déclaration

```
class CTimeBuffer: public CArrayLong
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayLong
      CTimeBuffer

Méthodes de Classe

Attributs	
Size	Définit la taille du buffer
Paramètres	
SetSymbolPeriod	Définit le symbole et la période
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer par son index.
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayLong

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2024, MetaQuotes Ltd.

Size

Définit la taille du buffer.

```
void  Size(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Définit le symbole et la période.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbole
   const ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Nouveau symbole.

period

[in] Nouvelle période (énumération ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du buffer par son index.

```
long  At(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

CTickVolumeBuffer

CTickVolumeBuffer est une classe permettant un accès simplifié aux volumes des ticks des barres de l'historique.

Description

La classe CTickVolumeBuffer fournit l'accès aux volumes des ticks des barres de l'historique.

Déclaration

```
class CTickVolumeBuffer: public CArrayLong
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayLong
      CTickVolumeBuffer

Méthodes de Classe

Attributs	
Size	Définit la taille du buffer
Paramètres	
SetSymbolPeriod	Définit le symbole et la période
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer par son index.
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayLong

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2024, MetaQuotes Ltd.

Size

Définit la taille du buffer.

```
void  Size(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Définit le symbole et la période.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbole
   const ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Nouveau symbole.

period

[in] Nouvelle période (énumération ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du buffer par son index.

```
long  At(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

CRealVolumeBuffer

CCloseBuffer est une classe permettant un accès simplifié aux volumes rééls des barres de l'historique.

Description

La classe CTickVolumeBuffer fournit l'accès aux volumes rééls des barres de l'historique.

Déclaration

```
class CRealVolumeBuffer: public CArrayLong
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayLong
      CRealVolumeBuffer

Méthodes de Classe

Attributs	
Size	Définit la taille du buffer
Paramètres	
SetSymbolPeriod	Définit le symbole et la période
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer par son index.
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayLong

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2024, MetaQuotes Ltd.

Size

Définit la taille du buffer.

```
void  Size(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Définit le symbole et la période.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbole
   const ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Nouveau symbole.

period

[in] Nouvelle période (énumération ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du buffer par son index.

```
long  At(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

CDoubleBuffer

CDoubleBuffer est une classe de base permettant un accès simplifié aux buffers de données de type double.

Description

La classe CDoubleBuffer fournit l'accès aux données de type double du buffer.

Déclaration

```
class CDoubleBuffer: public CArrayDouble
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayDouble
      CDoubleBuffer

Descendants directs

CCloseBuffer, CHighBuffer, CIndicatorBuffer, CLowBuffer, COpenBuffer

Méthodes de Classe

Attributs	
Size	Définit la taille du buffer
Paramètres	
SetSymbolPeriod	Définit le symbole et la période
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2024, MetaQuotes Ltd.

Size

Définit la taille du buffer.

```
void  Size(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Définit le symbole et la période.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbole
   const ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Nouveau symbole.

period

[in] Nouvelle période (énumération ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du buffer par son index.

```
double  At(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

COpenBuffer

COpenBuffer est une classe permettant un accès simplifié aux prix d'ouverture des barres dans l'historique.

Description

COpenBuffer est une classe permettant un accès simplifié aux prix d'ouverture des barres dans l'historique.

Déclaration

```
class COpenBuffer: public CDoubleBuffer
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        COpenBuffer

Méthodes de Classe

Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Méthodes héritées de la classe CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

CHighBuffer

CHighBuffer est une classe permettant un accès simplifié aux prix les plus hauts des barres de l'historique.

Description

La classe CHighBuffer fournit un accès aux prix les plus hauts des barres de l'historique.

Déclaration

```
class CHighBuffer: public CDoubleBuffer
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CHighBuffer

Méthodes de Classe

Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Méthodes héritées de la classe CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

CLowBuffer

CCloseBuffer est une classe permettant un accès simplifié aux prix les plus bas des barres dans l'historique.

Description

La classe CHighBuffer fournit un accès aux prix les plus bas des barres de l'historique.

Déclaration

```
class CLowBuffer: public CDoubleBuffer
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CLowBuffer

Méthodes de Classe

Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Méthodes héritées de la classe CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

CCloseBuffer

CCloseBuffer est une classe permettant un accès simplifié aux prix de clôture des barres dans l'historique.

Description

La classe CCloseBuffer fournit un accès aux prix de clôture des barres de l'historique.

Déclaration

```
class CCloseBuffer: public CDoubleBuffer
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CCloseBuffer

Méthodes de Classe

Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Méthodes héritées de la classe CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

CIndicatorBuffer

CIndicatorBuffer est une classe permettant un accès simplifié aux données du buffer de l'indicateur.

Description

La classe CIndicatorBuffer fournit un accès simplifié aux données du buffer de l'indicateur technique.

Déclaration

```
class CIndicatorBuffer: public CDoubleBuffer
```

Titre

```
#include <Indicators\Indicator.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CIndicatorBuffer

Méthodes de Classe

Attributs	
Offset	Retourne/Définit le décalage du buffer
Name	Retourne/Définit le nom du buffer
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer
Méthodes de Mise à Jour des Données	
Refresh	Met à jour le buffer
RefreshCurrent	Met à jour seulement la valeur courante

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayDouble

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Méthodes héritées de la classe CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

Offset

Retourne le décalage du buffer

```
int  Offset() const
```

Valeur de retour

Décalage du buffer.

Offset

Définit le décalage du buffer.

```
void  Offset(
   const int  offset      // décalage
   )
```

Paramètres

offset

[in] Nouveau décalage du buffer.

© 2000-2024, MetaQuotes Ltd.

Name

Retourne le nom du buffer.

```
string  Name() const
```

Valeur de retour

Nom du buffer.

Name

Définit le nom du buffer.

```
void  Name(
   const string  name      // nom
   )
```

Paramètres

name

[in] Nouveau nom du buffer.

© 2000-2024, MetaQuotes Ltd.

At

Retourne l'élément du buffer par son index.

```
double  At(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour le buffer en entier.

```
bool  Refresh(
   const int  handle,      // handle
   const int  num          // numéro du buffer
   )
```

Paramètres

handle

[in] Handle de l'indicateur.

num

[in] Index du buffer de l'indicateur.

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
bool  RefreshCurrent(
   const int  handle,      // handle de l'indicateur
   const int  num          // numéro du buffer
   )
```

Paramètres

handle

[in] Handle de l'indicateur.

num

[in] Numéro du buffer.

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2024, MetaQuotes Ltd.

CSeries

CSeries est la classe de base pour accéder aux séries de données de la Bibliothèque Standard.

Description

La classe CSeries fournit l'accès aux fonctions des séries de données MQL5 à tous ses descendants.

Déclaration

```
class CSeries: public CArrayObj
```

Titre

```
#include <Indicators\Series.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries

Descendants directs

CIndicator, CiRealVolume, CiSpread, CiTickVolume, CiTime, CPriceSeries

Méthodes de Classe

Attributs	
Name	Retourne le nom de la série de données ou de l'indicateur
BuffersTotal	Retourne le nombre de buffers d'une série de données ou d'un indicateur
Timeframe	Retourne le flag de la période de la série de données ou de l'indicateur
Symbol	Retourne le symbole de la série de données ou de l'indicateur
Period	Retourne la période de la série de données ou de l'indicateur
RefreshCurrent	Retourne/Définit le flag de mise à jour des données courantes
Méthodes d'Accès aux Données	
virtual BufferResize	Définit la taille du buffer de la série de données ou de l'indicateur
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour les données de la série de données ou de l'indicateur

© 2000-2024, MetaQuotes Ltd.

Attributs	
PeriodDescription	Retourne la période sous forme d'une chaîne de caractères

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2024, MetaQuotes Ltd.

Name

Retourne le nom de la série de données ou de l'indicateur

```
string  Name() const
```

Valeur de retour

Le nom de la série de données ou de l'indicateur

© 2000-2024, MetaQuotes Ltd.

BuffersTotal

Retourne le nombre de buffers d'une série de données ou d'un indicateur.

```
int  BuffersTotal() const
```

Valeur de retour

Le nombre de buffer de la série de données ou de l'indicateur.

Note

La série de données n'a qu'un seul buffer.

© 2000-2024, MetaQuotes Ltd.

Timeframe

Retourne le flag de la période de la série de données ou de l'indicateur.

```
int  Timeframe() const
```

Valeur de retour

Le flag de la période de la série de données ou de l'indicateur.

Note

C'est le flag de visibilité de certaines périodes.

© 2000-2024, MetaQuotes Ltd.

Symbol

Retourne le symbole de la série de données ou de l'indicateur.

```
string  Symbol() const
```

Valeur de retour

Le symbole de la série de données ou de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Period

Retourne la période de la série de données ou de l'indicateur.

```
ENUM_TIMEFRAMES  Period() const
```

Valeur de retour

La période (valeur de l'énumération ENUM_TIMEFRAMES) de la série de données ou de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Définit un flag pour mettre à jour les valeurs courantes de la série de données ou de l'indicateur.

```
string  RefreshCurrent(
   const bool  flag      // nouveau flag
   )
```

Paramètres

flag

[in] Nouveaux flag de visibilité.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

BufferSize

Retourne le nombre de données disponibles dans le buffer de la série de données ou de l'indicateur.

```
int  BufferSize() const
```

Valeur de retour

Nombre de données disponibles dans le buffer de la série de données ou de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Définit la taille du buffer de la série de données ou de l'indicateur.

```
virtual bool  BufferResize(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

Tous les buffers des série de données ou de l'indicateur ont la même taille.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour les données de la série de données ou de l'indicateur.

```
virtual void  Refresh(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Périodes à mettre à jour (flag).

© 2000-2024, MetaQuotes Ltd.

PeriodDescription

Retourne la représentation sous forme de chaîne de caractères de l'énumération ENUM_TIMEFRAMES spécifiée.

```
string  PeriodDescription(
   const int  val=0      // Valeur
   )
```

Paramètres

val=0

[in] Valeur à convertir.

Valeur de retour

La représentation sous forme de chaîne de caractères de l'énumération ENUM_TIMEFRAMES spécifiée.

Note

Si la valeur n'est pas spécifiée ou est égale à 0, retourne la période de la série de données ou de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

CPriceSeries

La classe CPriceSeries est la classe de base pour accéder aux données prix.

Description

La classe CSeries fournit un accès simplifié à tous ses descendants aux fonctions MQL5 permettant de travailler avec les prix.

Déclaration

```
class CPriceSeries: public CSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries

Descendants directs

CiClose, CiHigh, CiLow, CiOpen

Méthodes de Classe

Méthodes de Création	
virtual BufferResize	Définit la taille du buffer.
Méthodes d'Accès aux Données	
virtual GetData	Retourne l'élément du buffer spécifié par son index.
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour les séries de données
Méthodes de Recherche de Données	
virtual MinIndex	Retourne l'index du plus petit élément dans l'intervalle spécifié
virtual MinValue	Retourne la valeur et l'index de l'élément minimal du buffer spécifié dans un intervalle spécifié
virtual MaxIndex	Retourne l'index du plus grand élément dans l'intervalle spécifié

© 2000-2024, MetaQuotes Ltd.

Méthodes de Création	
virtual MaxValue	Retourne la valeur et l'index de l'élément maximal du buffer spécifié dans un intervalle spécifié

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

BufferResize

Définit la nouvelle taille du buffer.

```
virtual void  BufferResize(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2024, MetaQuotes Ltd.

GetData

Retourne l'élément du buffer spécifié par son index.

```
double  GetData(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

L'élément du buffer à l'index spécifié ou EMPTY_VALUE.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour les séries de données.

```
virtual void  Refresh(
   const int  flags=OBJ_ALL_PERIODS      // période flags
   )
```

Paramètres

flags=OBJ_ALL_PERIODS

[in] Périodes à mettre à jour (flag).

© 2000-2024, MetaQuotes Ltd.

MinIndex

Retourne l'index du plus petit élément dans l'intervalle spécifié.

```
virtual int  MinIndex(
   const int  start,      // index de départ
   const int  count       // nombre d'éléments à analyser
   ) const
```

Paramètres

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

Valeur de retour

L'index de l'élément minimal dans l'intervalle donné, ou -1 en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

MinValue

Retourne la valeur et l'index de l'élément minimal dans un intervalle spécifié.

```
virtual double  MinValue(
   const int   start,     // index de départ
   const int   count,     // nombre d'éléments à analyser
   int&        index      // référence à une variable pour l'index
   ) const
```

Paramètres

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

index

[out] Référence à une variable de type integer.

Valeur de retour

La valeur de l'élément minimal du buffer dans un intervalle spécifié, ou EMPTY_VALUE en cas d'erreur.

Note

L'index du plus petit élément est stocké dans la variable index.

© 2000-2024, MetaQuotes Ltd.

MaxIndex

Retourne l'index du plus grand élément dans l'intervalle spécifié.

```
virtual int  MaxIndex(
   const int  start,      // index de départ
   const int  count       // nombre d'éléments à analyser
   ) const
```

Paramètres

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

Valeur de retour

L'index de l'élément maximal du buffer spécifié dans l'intervalle donné, ou -1 en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

MaxValue

Retourne la valeur et l'index de l'élément maximal du buffer spécifié dans l'intervalle spécifié.

```
virtual double  MaxValue(
   const int   start,     // index de départ
   const int   count,     // nombre d'éléments à analyser
   int&        index      // référence à une variable pour l'index
   ) const
```

Paramètres

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

index

[out] Référence à une variable de type integer.

Valeur de retour

La valeur de l'élément maximal du buffer spécifié dans un intervalle spécifié, ou EMPTY_VALUE en cas d'erreur.

Note

L'index du plus grand élément est stocké dans la variable index.

© 2000-2024, MetaQuotes Ltd.

CIndicator

CIndicator est la classe de base des classes d'indicateurs techniques de la Bibliothèque Standard MQL.

Description

La classe CIndicator fournit un accès simplifié et à tous ses descendants aux fonctions des indicateurs techniques de l'API MQL5.

Déclaration

```
class CIndicator: public CSeries
```

Titre

```
#include <Indicators\Indicator.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator

Descendants directs

CiAC, CiAD, CiADX, CiADXWilder, CiAlligator, CiAMA, CiAO, CiATR, CiBands, CiBearsPower, CiBullsPower, CiBWMFI, CiCCI, CiChaikin, CiCustom, CiDEMA, CiDeMarker, CiEnvelopes, CiForce, CiFractals, CiFrAMA, CiGator, CiIchimoku, CiMA, CiMACD, CiMFI, CiMomentum, CiOBV, CiOsMA, CiRSI, CiRVI, CiSAR, CiStdDev, CiStochastic, CiTEMA, CiTriX, CiVIDyA, CiVolumes, CiWPR

Méthodes de Classe

Attributs	
Handle	Retourne le handle de l'indicateur.
Status	Retourne le statut de l'indicateur.
FullRelease	Définit le mode de libération du handle de l'indicateur.
Création	
Create	Crée l'indicateurs
BufferResize	Définit la nouvelle taille du buffer
Méthodes d'Accès aux Données	
GetData	Copie les données depuis le buffer de l'indicateur

© 2000-2024, MetaQuotes Ltd.

Attributs	
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données de l'indicateur
Recherche des Valeurs Min/Max	
Minimum	Retourne l'index de l'élément minimal du buffer spécifié dans un intervalle donné.
MinValue	Retourne la valeur et l'index de l'élément minimal du buffer spécifié dans un intervalle spécifié.
Maximum	Retourne l'index de l'élément maximal du buffer spécifié dans un intervalle donné.
MaxValue	Retourne la valeur et l'index de l'élément maximal du buffer spécifié dans un intervalle spécifié.
Conversion des Enumérations	
MethodDescription	Retourne la valeur de l'énumération ENUM_MA_METHOD sous forme de chaîne de caractères
PriceDescription	Retourne la valeur de l'énumération ENUM_APPLIED_PRICE sous forme de chaîne de caractères
VolumeDescription	Retourne la valeur de l'énumération ENUM_APPLIED_VOLUME sous forme de chaîne de caractères
Utilisation avec le graphique	
AddToChart	Ajoute l'indicateur au graphique
DeleteFromChart	Enlève l'indicateur du graphique

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Classes dérivées :

- CiAC
- CiAD
- CiADX
- CiADXWilder
- CiAlligator
- CiAMA
- CiAO
- CiATR
- CiBands
- CiBearsPower
- CiBullsPower
- CiBWMFI
- CiCCI
- CiChaikin
- CiDEMA
- CiDeMarker
- CiEnvelopes
- CiForce
- CiFractals
- CiFrAMA
- CiGator
- CiIchimoku
- CiMA
- CiMACD
- CiMFI
- CiMomentum
- CiOBV
- CiOsMA
- CiRSI
- CiRVI
- CiSAR
- CiStdDev
- CiStochastic
- CiTEMA
- CiTriX
- CiVIDyA
- CiVolumes

© 2000-2024, MetaQuotes Ltd.

- CiWPR

© 2000-2024, MetaQuotes Ltd.

Handle

Retourne le handle de l'indicateur.

```
int  Handle() const
```

Valeur de retour

Handle de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Status

Retourne le statut de l'indicateur.

```
string  Status() const
```

Valeur de retour

Le statut de création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

FullRelease

Définit le mode de libération du handle de l'indicateur.

```
void  FullRelease(
   const bool  flag=true      // flag
   )
```

Paramètres

flag

[in] Nouvelle valeur du paramètre the handle release flag.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur with specified parameters.

```
bool  Create(
   const string            symbol,          // symbole
   const ENUM_TIMEFRAMES   period,          // période
   const ENUM_INDICATOR    type,            // type
   const int               num_params,      // nombre de paramètres
   const MqlParam&         params[]         // référence sur le tableau des paramètres
   )
```

Paramètres

symbol

[in] Nom du symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

type

[in] Type de l'indicateur (énumération ENUM_INDICATOR).

num_params

[in] Nombre de paramètres de l'indicateur.

params

[in] Référence sur le tableau de paramètres de l'indicateur.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Définit la taille du buffer de l'indicateur.

```
virtual bool  BufferResize(
   const int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

Tous les buffers de l'indicateur ont la même taille.

© 2000-2024, MetaQuotes Ltd.

BarsCalculated

Retourne le nombre de données calculées pour l'indicateur.

```
int  BarsCalculated() const;
```

Valeur de Retour

Retourne le nombre de données calculées dans le buffer de l'indicateur, ou -1 en cas d'erreur (données pas encore calculées).

© 2000-2024, MetaQuotes Ltd.

GetData

Retourne l'élément spécifié du buffer spécifié de l'indicateur. La méthode Refresh() devrait être appelée pour pouvoir travailler avec les données les plus récentes avant d'utiliser la méthode GetData.

```
double  GetData(
   const int  buffer_num,      // numéro du buffer
   const int  index            // index de l'élément
   ) const
```

Paramètres

buffer_num

[in] Numéro du buffer.

index

[in] Index de l'élément.

Valeur de retour

En cas de succès, retourne la valeur numérique de l'élément, ou EMPTY_VALUE en cas d'erreur.

GetData

Retourne les données du buffer de l'indicateur à partir d'une position donnée et le nombre de données à retourner.

```
int  GetData(
   const int      start_pos,      // position
   const int      count,          // nombre d'éléments demandés
   const int      buffer_num,     // numéro du buffer
   double&        buffer[]        // tableau destination pour les données
   ) const
```

Paramètres

start_pos

[in] Position de départ du buffer de l'indicateur.

count

[in] Nombre d'éléments à retourner.

buffer_num

[in] Numéro du buffer de l'indicateur.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

En cas de succès, retourne le nombre d'éléments reçus du buffer spécifié de l'indicateur, sinon -1.

GetData

© 2000-2024, MetaQuotes Ltd.

Retourne les données du buffer de l'indicateur à partir d'une heure de départ et le nombre de données à retourner.

```
int  GetData(
   const datetime  start_time,      // date/heure de départ
   const int       count,           // nombre d'éléments à retourner
   const int       buffer_num,      // numéro du buffer
   double&         buffer[]         // tableau destination pour les données
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer_num

[in] Numéro du buffer de l'indicateur.

buffer

[in] Référence vers le tableau destination.

Valeur de retour

En cas de succès, retourne le nombre d'éléments reçus du buffer spécifié de l'indicateur, sinon -1.

GetData

Retourne les données du buffer de l'indicateur à partir d'une heure de départ et de fin et le nombre de données à retourner.

```
int  GetData(
   const datetime  start_time,      // date/heure de départ
   const datetime  stop_time,       // date/heure de fin
   const int       buffer_num,      // numéro du buffer
   double&         buffer[]         // tableau destination des données
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer_num

[in] Numéro du buffer de l'indicateur.

buffer

[in] Référence vers le tableau destination.

© 2000-2024, MetaQuotes Ltd.

Valeur de retour

En cas de succès, retourne le nombre d'éléments reçus du buffer spécifié de l'indicateur, sinon -1.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour les données de l'indicateur. Il est recommendé d'appeler cette fonction avant d'utiliser GetData().

```
virtual void  Refresh(
   int  flags=OBJ_ALL_PERIODS      // flags
   )
```

Paramètres

flags=OBJ_ALL_PERIODS

[in] Flags des périodes à mettre à jour.

© 2000-2024, MetaQuotes Ltd.

Minimum

Retourne l'index de l'élément minimal du buffer spécifié dans un intervalle donné.

```
int  Minimum(
   const int  buffer_num,      // numéro du buffer
   const int  start,           // index de départ
   const int  count            // nombre d'éléments à procéder
   ) const
```

Paramètres

buffer_num

[in] Numéro du buffer dans lequel la valeur doit être recherchée.

start

[in] Index de départ de la recherche.

count

[in] Nombre d'éléments à rechercher.

Valeur de retour

L'index de l'élément minimal du buffer spécifié dans un intervalle donné.

© 2000-2024, MetaQuotes Ltd.

MinValue

Retourne la valeur et l'index de l'élément minimal du buffer spécifié dans un intervalle spécifié.

```
double  MinValue(
   const int   buffer_num,      // numéro du buffer
   const int   start,           // index de départ
   const int   count,           // nombre d'éléments à traiter
   int&        index            // référence
   ) const
```

Paramètres

buffer_num

[in] Numéro du buffer dans lequel la valeur doit être recherchée.

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

index

[out] Référence à une variable de type int pour l'index de l'élément minimal.

Valeur de retour

La valeur de l'élément minimal du buffer spécifié dans un intervalle spécifié.

Note

L'index du plus petit élément du buffer est stocké dans la variable index, passée par référence.

© 2000-2024, MetaQuotes Ltd.

Maximum

Retourne l'index de l'élément maximal du buffer spécifié dans un intervalle donné.

```
int  Maximum(
   const int  buffer_num,      // numéro du buffer
   const int  start,           // index de départ
   const int  count            // nombre d'éléments à procéder
   ) const
```

Paramètres

buffer_num

[in] Numéro du buffer dans lequel la valeur doit être recherchée.

start

[in] Index de départ de la recherche.

count

[in] Nombre d'éléments à rechercher.

Valeur de retour

Index de l'élément maximal du buffer spécifié dans un intervalle donné.

© 2000-2024, MetaQuotes Ltd.

MaxValue

Retourne la valeur et l'index de l'élément maximal du buffer spécifié dans un intervalle spécifié.

```
double  MaxValue(
   const int   buffer_num,      // numéro du buffer
   const int   start,           // index de départ
   const int   count,           // nombre d'éléments à traiter
   int&        index            // référence
   ) const
```

Paramètres

buffer_num

[in] Numéro du buffer dans lequel la valeur doit être recherchée.

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

index

[out] Référence à une variable de type int pour l'index de l'élément maximal.

Valeur de retour

La valeur de l'élément maximal du buffer spécifié dans un intervalle spécifié.

Note

L'index du plus grand élément du buffer est stocké dans la variable index, passée par référence.

© 2000-2024, MetaQuotes Ltd.

MethodDescription

La fonction retourne la valeur de l'énumération ENUM_MA_METHOD sous forme d'une chaîne de caractères.

```
string  MethodDescription(
   const int  val      // Valeur
   ) const
```

Paramètres

val

[in] Valeur de l'énumération ENUM_MA_METHOD.

Valeur de retour

La valeur de l'énumération ENUM_MA_METHOD sous forme d'une chaîne de caractères.

© 2000-2024, MetaQuotes Ltd.

PriceDescription

La fonction retourne la valeur de l'énumération ENUM_APPLIED_PRICE sous forme d'une chaîne de caractères.

```
string  PriceDescription(
   const int  val      // Valeur
   ) const
```

Paramètres

val

[in] Valeur de l'énumération ENUM_APPLIED_PRICE.

Valeur de retour

La valeur de l'énumération ENUM_APPLIED_PRICE sous forme d'une chaîne de caractères.

© 2000-2024, MetaQuotes Ltd.

VolumeDescription

La fonction retourne la valeur de l'énumération ENUM_APPLIED_VOLUME sous forme d'une chaîne de caractères.

```
string  VolumeDescription(
   const int  val      // Valeur
   ) const
```

Paramètres

val

[in] Valeur de l'énumération ENUM_APPLIED_VOLUME.

Valeur de retour

La valeur de l'énumération ENUM_APPLIED_VOLUME sous forme d'une chaîne de caractères.

© 2000-2024, MetaQuotes Ltd.

AddToChart

Ajoute l'indicateur au graphique.

```
bool  AddToChart(
   const long  chart,      // Identifiant du graphique
   const int  subwin       // sous-fenêtre du graphique
   )
```

Paramètres

chart

[in] Identifiant du graphique.

subwin

[in] Sous-fenêtre du graphique.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

DeleteFromChart

Enlève l'indicateur du graphique.

```
bool  DeleteFromChart(
   const long  chart,      // Identifiant du graphique
   const int  subwin       // sous-fenêtre du graphique
   )
```

Paramètres

chart

[in] Identifiant du graphique.

subwin

[in] Sous-fenêtre du graphique.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

CIndicators

La classe CIndicators permmet d'avoir une liste d'instances de classes de séries de données et d'indicateurs techniques.

Description

La classe CIndicators fournit la création des instances de classes des indicateurs techniques, leur stockage et leur gestion (synchronisation des données, gestion du handle et de la mémoire).

Déclaration

```
class CIndicators: public CArrayObj
```

Titre

```
#include <Indicators\Indicators.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée l'indicateur technique
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données de tous les indicateurs techniques de la collection.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés.

```
CIndicator*  Create(
   const string            symbol,      // Symbole name
   const ENUM_TIMEFRAMES   period,      // Période
   const ENUM_INDICATOR    type,        // Type de l'indicateur
   const int               count,       // nombre de paramètres
   const MqlParam&         params       // Référence sur le tableau des paramètres
   )
```

Paramètres

symbol

[in] Nom du symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

type

[in] Type de l'indicateur (ENUM_INDICATOR).

count

[in] Nombre de paramètres de l'indicateur.

params

[in] Référence sur le tableau de paramètres de l'indicateur.

Valeur de retour

En cas de succès, retourne la référence sur l'indicateur créé, et NULL si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour les données de tous les indicateurs techniques de la collection.

```
int  Refresh()
```

Valeur de retour

Retourne les flags des périodes mises à jour.

© 2000-2024, MetaQuotes Ltd.

Classes des séries de données

Cet ensemble de chapitres contient les détails techniques des classes des séries de données de la Bibliothèque Standard MQL5 et les descriptions de ses composants principaux.

Classe	Description
CiSpread	Fournit un accès aux données historiques des spreads
CiTime	Fournit un accès aux données historiques des heures d'ouverture des barres de l'historique
CiTickVolume	Fournit un accès aux données historiques des volumes des ticks des barres de l'historique
CiRealVolume	Fournit un accès aux données historiques des volumes réels des barres de l'historique
CiOpen	Fournit un accès aux données historiques des prix d'ouverture (open) des barres de l'historique
CiHigh	Fournit un accès aux données historiques des prix plus hauts (high) des barres de l'historique
CiLow	Fournit un accès aux données historiques des prix plus bas (low) des barres de l'historique
CiClose	Fournit un accès aux données historiques des prix de clôture (close) des barres de l'historique

© 2000-2024, MetaQuotes Ltd.

CiSpread

CiSpread est une classe permettant d'accéder aux spreads des barres de l'historique.

Description

La classe CiSpread fournit l'accès aux spreads des barres de l'historique.

Déclaration

```
class CiSpread: public CSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

- CObject
  - CArray
    - CArrayObj
      - CSeries
        - CiSpread

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
BufferResize	Définit la taille du buffer
Méthodes d'Accès aux Données	
GetData	Retourne la donnée
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux spreads de l'historique.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Définit la nouvelle taille de la série.

```
virtual void  BufferResize(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2024, MetaQuotes Ltd.

GetData

Retourne l'élément de la série par son index.

```
int  GetData(
   int  index       // index
   ) const
```

Paramètres

index

[in] Index de l'élément à retourner.

Valeur de retour

L'élément du buffer de la série de données, ou 0.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,      // position de départ
   int   count,          // nombre d'éléments à récupérer
   int&  buffer          // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   int&      buffer           // tableau destination
   ) const
```

Paramètres

start_time

© 2000-2024, MetaQuotes Ltd.

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

```
int  GetData(
   datetime  start_time,      // heure de départ
   datetime  stop_time,       // heure de fin
   int&      buffer           // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour les données de la série de données.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Paramètres

flags

[in] Flags de la période.

© 2000-2024, MetaQuotes Ltd.

CiTime

CiTime est une classe permettant d'accéder aux heures d'ouverture des barres de l'historique.

Description

La classe CiTime fournit l'accès aux prix d'ouverture des barres de l'historique.

Déclaration

```
class CiTime: public CSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CiTime

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
BufferResize	Définit la taille du buffer
Méthodes d'Accès aux Données	
GetData	Retourne la donnée
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux heures d'ouverture des barres de l'historique.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Définit la nouvelle taille de la série.

```
virtual void  BufferResize(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2024, MetaQuotes Ltd.

GetData

Retourne l'élément de la série par son index.

```
datetime  GetData(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément à retourner.

Valeur de retour

L'élément du buffer de la série de données, ou 0.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,      // position de départ
   int   count,          // nombre d'éléments à récupérer
   long& buffer          // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   long&     buffer           // tableau destination
   ) const
```

Paramètres

start_time

© 2000-2024, MetaQuotes Ltd.

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

```
int  GetData(
   datetime  start_time,      // heure de départ
   datetime  stop_time,       // heure de fin
   long&     buffer           // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour les données de la série de données.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Paramètres

flags

[in] Flags de la période.

© 2000-2024, MetaQuotes Ltd.

CiTickVolume

CiTickVolume est une classe permettant un accès simplifié aux volumes des ticks des barres de l'historique.

Description

La classe CiTickVolume fournit l'accès aux volumes des ticks des barres de l'historique.

Déclaration

```
class CiTickVolume: public CSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CiTickVolume

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
BufferResize	Définit la taille du buffer
Méthodes d'Accès aux Données	
GetData	Retourne la donnée
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux volumes des ticks des barres de l'historique.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Définit la nouvelle taille de la série.

```
virtual void  BufferResize(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2024, MetaQuotes Ltd.

GetData

Retourne l'élément de la série par son index.

```
long  GetData(
   int  index       // index
   ) const
```

Paramètres

index

[in] Index de l'élément à retourner.

Valeur de retour

L'élément du buffer de la série de données, ou 0.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,      // position de départ
   int   count,          // nombre d'éléments à récupérer
   long& buffer          // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   long&     buffer           // tableau destination
   ) const
```

Paramètres

start_time

© 2000-2024, MetaQuotes Ltd.

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

```
int  GetData(
   datetime  start_time,      // heure de départ
   datetime  stop_time,       // heure de fin
   long&     buffer           // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour les données de la série de données.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Paramètres

flags

[in] Flags de la période.

© 2000-2024, MetaQuotes Ltd.

CiRealVolume

CiRealVolume est une classe permettant un accès simplifié aux volumes rééls des barres de l'historique.

Description

La classe CiRealVolume fournit l'accès aux volumes rééls des barres de l'historique.

Déclaration

```
class CiRealVolume: public CSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CiRealVolume

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
BufferResize	Définit la taille du buffer
Méthodes d'Accès aux Données	
GetData	Retourne la donnée
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux volumes réels des barres de l'historique.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Définit la nouvelle taille de la série.

```
virtual void  BufferResize(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2024, MetaQuotes Ltd.

GetData

Retourne l'élément de la série par son index.

```
datetime  GetData(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément à retourner.

Valeur de retour

L'élément du buffer de la série de données, ou 0.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,      // position de départ
   int   count,          // nombre d'éléments à récupérer
   long& buffer          // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   long&     buffer           // tableau destination
   ) const
```

Paramètres

start_time

© 2000-2024, MetaQuotes Ltd.

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

```
int  GetData(
   datetime  start_time,      // heure de départ
   datetime  stop_time,       // heure de fin
   long&     buffer           // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour les données de la série de données.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Paramètres

flags

[in] Flags de la période.

© 2000-2024, MetaQuotes Ltd.

CiOpen

CiOpen est une classe permettant d'accéder aux prix d'ouverture (open) des barres de l'historique.

Description

La classe CiOpen permet d'accéder aux prix d'ouverture (open) des barres de l'historique.

Déclaration

```
class CiOpen: public CPriceSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiOpen

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
Méthodes d'Accès aux Données	
GetData	Retourne la donnée

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2024, MetaQuotes Ltd.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux prix d'ouverture (open) des barres de l'historique.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2024, MetaQuotes Ltd.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,      // position de départ
   int   count,          // nombre d'éléments à récupérer
   double& buffer        // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   double&     buffer         // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

© 2000-2024, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // heure de départ
   datetime  stop_time,       // heure de fin
   double&   buffer           // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

CiHigh

CiHigh est une classe permettant d'accéder aux prix plus hauts (high) des barres de l'historique.

Description

La classe CHigh fournit un accès aux prix les plus hauts des barres de l'historique.

Déclaration

```
class CiHigh: public CPriceSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiHigh

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
Méthodes d'Accès aux Données	
GetData	Retourne la donnée

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2024, MetaQuotes Ltd.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux prix plus hauts (high) des barres de l'historique.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2024, MetaQuotes Ltd.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,      // position de départ
   int   count,          // nombre d'éléments à récupérer
   double& buffer        // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   double&     buffer         // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

© 2000-2024, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // heure de départ
   datetime  stop_time,       // heure de fin
   double&   buffer           // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

CiLow

CiLow est une classe permettant d'accéder aux prix plus bas (low) des barres de l'historique.

Description

La classe CiLow fournit un accès aux prix les plus bas des barres de l'historique.

Déclaration

```
class CiLow: public CPriceSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiLow

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
Méthodes d'Accès aux Données	
GetData	Retourne la donnée

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2024, MetaQuotes Ltd.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux prix plus bas (low) des barres de l'historique.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2024, MetaQuotes Ltd.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,      // position de départ
   int   count,          // nombre d'éléments à récupérer
   double& buffer        // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   double&     buffer         // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

© 2000-2024, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // heure de départ
   datetime  stop_time,       // heure de fin
   double&   buffer           // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

CiClose

CiClose est une classe permettant d'accéder aux prix de clôture des barres de l'historique.

Description

La classe CiClose fournit un accès aux prix de clôture des barres de l'historique.

Déclaration

```
class CiClose: public CPriceSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiClose

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
Méthodes d'Accès aux Données	
GetData	Retourne la donnée

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2024, MetaQuotes Ltd.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux prix de clôture des barres de l'historique.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2024, MetaQuotes Ltd.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,      // position de départ
   int   count,          // nombre d'éléments à récupérer
   double& buffer        // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   double&     buffer         // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

© 2000-2024, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // heure de départ
   datetime  stop_time,       // heure de fin
   double&   buffer           // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

Classes d'Indicateurs de Tendance

Cet ensemble de chapitres contient les détails techniques des classes des indicateurs de Tendance de la Bibliothèque Standard MQL5 et les descriptions de ses composants principaux.

Classe/groupe	Description
CiADX	Average Directional Index
CiADXWilder	Average Directional Index par Welles Wilder
CiBands	Bandes de Bollinger®
CiEnvelopes	Enveloppes
CiIchimoku	Ichimoku Kinko Hyo
CiMA	Moyenne Mobile
CiSAR	Système Parabolic Stop And Reverse
CiStdDev	Standard Deviation
CiDEMA	Double Exponential Moving Average
CiTEMA	Triple Exponential Moving Average
CiFrAMA	Fractal Adaptive Moving Average
CiAMA	Adaptive Moving Average
CiVIDyA	Variable Index Dynamic Average

© 2000-2024, MetaQuotes Ltd.

CiADX

La classe CiADX permet d'utiliser l'indicateur technique Average Directional Index.

Description

La classe CiADX fournit la création, la configuration et l'accès aux données de l'indicateur Average Directional Index.

Déclaration

```
class CiADX: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiADX

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne principale
Plus	Retourne l'élément du buffer de la ligne +DI
Minus	Retourne l'élément du buffer de la ligne -DI
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Plus

Retourne l'élément du buffer de la ligne +DI de l'index spécifié.

```
double  Plus(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne +DI de l'index spécifié, ou ***<t3>EMPTY_VALUE s'il n'y a aucune donnée correcte.</t3>

© 2000-2024, MetaQuotes Ltd.

Minus

Retourne l'élément du buffer de la ligne -DI de l'index spécifié.

```
double  Minus(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne -DI de l'index spécifié, ou EMPTY_VALUE s'il n'y a aucune donnée correcte.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ADX for CiADX).

© 2000-2024, MetaQuotes Ltd.

CiADXWilder

La classe CiADX permet d'utiliser l'indicateur technique Average Directional Index de Welles Wilder.

Description

La classe CiADXWilder fournit la création, la configuration et l'accès aux données de l'indicateur Average Directional Index de Welles Wilder.

Déclaration

```
class CiADXWilder: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiADXWilder

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne principale
Plus	Retourne l'élément du buffer de la ligne +DI
Minus	Retourne l'élément du buffer de la ligne -DI
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Plus

Retourne l'élément du buffer de la ligne +DI de l'index spécifié.

```
double  Plus(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne +DI de l'index spécifié, ou ***<t3>EMPTY_VALUE s'il n'y a aucune donnée correcte.</t3>

© 2000-2024, MetaQuotes Ltd.

Minus

Retourne l'élément du buffer de la ligne -DI de l'index spécifié.

```
double  Minus(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne -DI de l'index spécifié, ou EMPTY_VALUE s'il n'y a aucune donnée correcte.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ADXW for CiADXWilder).

© 2000-2024, MetaQuotes Ltd.

CiBands

La classe CiBands permet d'utiliser l'indicateur technique des Bandes de Bollinger®.

Description

La classe CiBands fournit la création, la configuration et l'accès aux données de l'indicateur des Bandes de Bollinger.

Déclaration

```
class CiBands: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBands

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
MaShift	Retourne le décalage horizontal
Deviation	Retourne la déviation
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Base	Retourne l'élément du buffer de la ligne de base
Upper	Retourne l'élément du buffer de la ligne supérieure
Lower	Retourne l'élément du buffer de la ligne inférieure
Entrée/Sortie	

© 2000-2024, MetaQuotes Ltd.

Attributs	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaShift

Retourne le décalage horizontal de l'indicateur.

```
int  MaShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Deviation

Retourne la déviation.

```
double  Deviation() const
```

Valeur de retour

Retourne la déviation définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage
   double           deviation,     // Déviation
   int              applied        // prix à appliquer ou handle
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal de l'indicateur.

deviation

[in] Déviation.

applied

[in] Type de volume à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Base

Retourne l'élément du buffer de la ligne de base de l'index spécifié.

```
double  Base(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne de Base de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Upper

Retourne l'élément du buffer de la ligne supérieure de l'index spécifié.

```
double  Upper(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne supérieure de l'index spécifié, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Lower

Retourne l'élément du buffer de la ligne inférieure de l'index spécifié.

```
double  Lower(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne inférieure de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_BANDS for CiBands).

© 2000-2024, MetaQuotes Ltd.

CiEnvelopes

CiEnvelopes est la classe permettant d'utiliser l'indicateur technique Enveloppes.

Description

La classe CiEnvelopes fournit la création, la configuration et l'accès aux données de l'indicateur Enveloppes.

Déclaration

```
class CiEnvelopes: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiEnvelopes

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
MaShift	Retourne le décalage horizontal
MaMethod	Retourne la méthode de lissage
Deviation	Retourne la déviation
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Upper	Retourne l'élément du buffer de la ligne supérieure
Lower	Retourne l'élément du buffer de la ligne inférieure
Entrée/Sortie	

© 2000-2024, MetaQuotes Ltd.

Attributs	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaShift

Retourne le décalage horizontal de l'indicateur.

```
int  MaShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Deviation

Retourne la valeur de la déviation.

```
double  Deviation() const
```

Valeur de retour

Retourne la valeur de déviation définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage horizontal
   ENUM_MA_METHOD   ma_method,     // Méthode de lissage
   int              applied,       // prix type ou handle to apply
   double           deviation      // Déviation
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Prix type of handle to apply.

deviation

[in] Déviation.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Upper

Retourne l'élément du buffer de la ligne supérieure de l'index spécifié.

```
double  Upper(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne supérieure de l'index spécifié, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Lower

Retourne l'élément du buffer de la ligne inférieure de l'index spécifié.

```
double  Lower(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne inférieure de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ENVELOPES for CiEnvelopes).

© 2000-2024, MetaQuotes Ltd.

CiIchimoku

La classe CiIchimoku permet d'utiliser l'indicateur technique Ichimoku Kinko Hyoe.

Description

La classe CiIchimoku fournit la création, la configuration et l'accès aux données de l'indicateur Ichimoku Kinko Hyo.

Déclaration

```
class CiIchimoku: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
- CArray
  - CArrayObj
    - CSeries
      - CIndicator
        - CiIchimoku

Méthodes de Classe

Attributs	
TenkanSenPeriod	Retourne la période TenkanSen
KijunSenPeriod	Retourne la période KijunSen
SenkouSpanBPeriod	Retourne la période SenkouSpanB
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
TenkanSen	Retourne l'élément du buffer de la ligne TenkanSen
KijunSen	Retourne l'élément du buffer de la ligne KijunSen
SenkouSpanA	Retourne l'élément du buffer de la ligne SenkouSpanA
SenkouSpanB	Retourne l'élément du buffer de la ligne SenkouSpanB

© 2000-2024, MetaQuotes Ltd.

Attributs	
ChinkouSpan	Retourne l'élément du buffer de la ligne ChikouSpan
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

TenkanSenPeriod

Retourne la période TenkanSen.

```
int  TenkanSenPeriod() const
```

Valeur de retour

Retourne la période TenkanSen, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

KijunSenPeriod

Retourne la période KijunSen

```
int  KijunSenPeriod() const
```

Valeur de retour

Retourne la période KijunSen, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

SenkouSpanBPeriod

Retourne la période SenkouSpanB.

```
int  SenkouSpanBPeriod() const
```

Valeur de retour

Retourne la période SenkouSpanB, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,            // Symbole
   ENUM_TIMEFRAMES  period,            // Période
   int              tenkan_sen,        // Période of TenkanSen
   int              kijun_sen,         // Période of KijunSen
   int              senkou_span_b      // Période of SenkouSpanB
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

tenkan_sen

[in] Période de TenkanSen.

kijun_sen

[in] Période de KijunSen.

senkou_span_b

[in] Période de SenkouSpanB.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

TenkanSen

Retourne l'élément du buffer de la ligne TenkanSen de l'index spécifié.

```
double  TenkanSen(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne TenkanSen de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

KijunSen

Retourne l'élément du buffer de la ligne KijunSen de l'index spécifié.

```
double  KijunSen(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne KijunSen de l'index spécifié, ou ***<t3>EMPTY_VALUE s'il n'y a aucune donnée correcte.</t3>

© 2000-2024, MetaQuotes Ltd.

SenkouSpanA

Retourne l'élément du buffer de la ligne SenkouSpanA de l'index spécifié.

```
double  SenkouSpanA(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne SenkouSpanA de l'index spécifié, ou ***<t3>EMPTY_VALUE s'il n'y a aucune donnée correcte.</t3>

© 2000-2024, MetaQuotes Ltd.

SenkouSpanB

Retourne l'élément du buffer de la ligne SenkouSpanB de l'index spécifié.

```
double  SenkouSpanB(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne SenkouSpanB de l'index spécifié, ou ***<t3>EMPTY_VALUE s'il n'y a aucune donnée correcte.</t3>

© 2000-2024, MetaQuotes Ltd.

ChinkouSpan

Retourne l'élément du buffer de la ligne ChinkouSpan de l'index spécifié.

```
double  ChinkouSpan(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne ChinkouSpan de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ICHIMOKU for Cilchimoku).

© 2000-2024, MetaQuotes Ltd.

CiMA

La classe CiMA permet d'utiliser l'indicateur technique Moving Average (Moyenne Mobile).

Description

La classe CiMA fournit la création, la configuration et l'accès aux données de l'indicateur Moving Average (Moyenne Mobile).

Déclaration

```
class CiMA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMA

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
MaShift	Retourne le décalage horizontal
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaShift

Retourne le décalage horizontal de l'indicateur.

```
int  MaShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage (valeur de l'énumération ENUM_MA_METHOD), définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           string,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage horizontal
   ENUM_MA_METHOD   ma_method,     // Méthode de lissage
   int              applied        // prix type of handle to apply
   )
```

Paramètres

string

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_MA for CiMA).

© 2000-2024, MetaQuotes Ltd.

CiSAR

La classe CiSAR permet d'utiliser l'indicateur technique Parabolic Stop And Reverse System.

Description

La classe CiSAR fournit la création, la configuration et l'accès aux données de l'indicateur Parabolic Stop And Reverse System.

Déclaration

```
class CiSAR: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiSAR

Méthodes de Classe

Attributs	
SarStep	Retourne le pas d'accroissement de la vélocité
Maximum	Retourne le coefficient de suivi du prix
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

SarStep

Retourne le pas d'accroissement de la vélocité (coefficient d'accélération).

```
double  SarStep() const
```

Valeur de retour

Le pas d'accroissement de la vélocité, défini à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Maximum

Retourne le coefficient de suivi du prix.

```
double  Maximum() const
```

Valeur de retour

Le coefficient de suivi du prix, défini à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,       // Symbole
   ENUM_TIMEFRAMES  period,       // Période
   double           step,         // Pas
   double           maximum       // Coefficient
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

step

[in] Pas d'accroissement de la vélocité.

maximum

[in] Prix following coefficient.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_SAR for CiSAR).

© 2000-2024, MetaQuotes Ltd.

CiStdDev

La classe CiStdDev permet d'utiliser l'indicateur technique Standard Deviation.

Description

La classe CiStdDev fournit la création, la configuration et l'accès aux données de l'indicateur technique Standard Deviation.

Déclaration

```
class CiStdDev: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiStdDev

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
MaShift	Retourne le décalage horizontal
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaShift

Retourne le décalage horizontal de l'indicateur.

```
int  MaShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage (valeur de l'énumération ***<t1></t1><li2>ENUM_MA_METHOD</li2><t4>), définie à la création de l'indicateur.</t4>

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage horizontal
   ENUM_MA_METHOD   ma_method,     // Méthode de lissage
   int              applied        // type de prix ou handle à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_STDDEV for CiStdDev).

© 2000-2024, MetaQuotes Ltd.

CiDEMA

La classe CiDEMA permet d'utiliser l'indicateur technique Double Exponential Moving Average.

Description

La classe CiDEMA fournit la création, la configuration et l'accès aux données de l'indicateur Double Exponential Moving Average.

Déclaration

```
class CiDEMA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiDEMA

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
IndShift	Retourne le décalage horizontal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

IndShift

Retourne le décalage horizontal de l'indicateur.

```
int  IndShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           string,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ind_shift,     // Décalage
   int              applied        // prix type of handle to apply
   )
```

Paramètres

string

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ind_shift

[in] Décalage horizontal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_DEMA for CiDEMA).

© 2000-2024, MetaQuotes Ltd.

CiTEMA

La classe CiTEMA permet d'utiliser l'indicateur technique Triple Exponential Moving Average.

Description

La classe CiTEMA fournit la création, la configuration et l'accès aux données de l'indicateur Triple Exponential Moving Average.

Déclaration

```
class CiTEMA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiTEMA

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
IndShift	Retourne le décalage horizontal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

IndShift

Retourne le décalage horizontal de l'indicateur.

```
int  IndShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage
   int              applied        // prix type of handle to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_TEMA for CiTEMA).

© 2000-2024, MetaQuotes Ltd.

CiFrAMA

La classe CiFrAMA permet d'utiliser l'indicateur technique Fractal Adaptive Moving Average.

Description

La classe CiFrAMA fournit la création, la configuration et l'accès aux données de l'indicateur Fractal Adaptive Moving Average.

Déclaration

```
class CiFrAMA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiFrAMA

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
IndShift	Retourne le décalage horizontal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

IndShift

Retourne le décalage horizontal de l'indicateur.

```
int  IndShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage
   int              applied        // prix type of handle to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_FRAMA for CiFrAMA).

© 2000-2024, MetaQuotes Ltd.

CiAMA

La classe CiTriX permet d'utiliser l'indicateur technique Adaptive Moving Average.

Description

La classe CiAMA fournit la création, la configuration et l'accès aux données de l'indicateur Adaptive Moving Average.

Déclaration

```
class CiAMA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiAMA

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
FastEmaPeriod	Retourne la période de lissage de la moyenne mobile rapide.
SlowEmaPeriod	Retourne la période de lissage de la moyenne mobile lente.
IndShift	Retourne le décalage horizontal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

FastEmaPeriod

Retourne la période de lissage de la moyenne mobile rapide.

```
int  FastEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile rapide, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

SlowEmaPeriod

Retourne la période de lissage de la moyenne mobile lente.

```
int  SlowEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile lente, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

IndShift

Retourne le décalage horizontal de l'indicateur.

```
int  IndShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           string,              // Symbole
   ENUM_TIMEFRAMES  period,              // Période
   int              ma_period,           // Période de lissage
   int              fast_ema_period,     // Période de la moyenne mobile rapide
   int              slow_ema_period,     // Période de la moyenne mobile lente
   int              ind_shift,           // Décalage
   int              applied              // prix type ou handle to apply
   )
```

Paramètres

string

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

fast_ema_period

[in] Période de la moyenne mobile rapide.

slow_ema_period

[in] Période de la moyenne mobile lente.

ind_shift

[in] Décalage horizontal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_AMA for CiAMA).

© 2000-2024, MetaQuotes Ltd.

CiVIDyA

La classe CiVIDyA permet d'utiliser l'indicateur technique Variable Index Dynamic Average.

Description

La classe CiVIDyA fournit la création, la configuration et l'accès aux données de l'indicateur ariable Index Dynamic Average.

Déclaration

```
class CiVIDyA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiVIDyA

Méthodes de Classe

Attributs	
CmoPeriod	Retourne la période du Momentum
EmaPeriod	Retourne la période de lissage de la moyenne mobile.
IndShift	Retourne le décalage horizontal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

CmoPeriod

Retourne la période du Momentum.

```
int  CmoPeriod() const
```

Valeur de retour

Retourne la période du Momentum, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

EmaPeriod

Retourne la période de lissage de la moyenne mobile.

```
int  EmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

IndShift

Retourne le décalage horizontal de l'indicateur.

```
int  IndShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,         // Symbole
   ENUM_TIMEFRAMES  period,         // Période
   int              cmo_period,     // Période du Momentum
   int              ema_period,     // Période de lissage
   int              ind_shift,      // Décalage
   int              applied         // prix type ou handle to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

cmo_period

[in] Période du Momentum.

ema_period

[in] Période de lissage.

ind_shift

[in] Décalage horizontal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_VIDYA for CiVIDyA).

© 2000-2024, MetaQuotes Ltd.

Oscillateurs

Cet ensemble de chapitres contient les détails techniques des classes d'Oscillateurs de la Bibliothèque Standard MQL5 et les descriptions de ses composants principaux.

Classe/groupe	Description
CiATR	Average True Range
CiBearsPower	Bears Power
CiBullsPower	Bulls Power
CiCCI	Commodity Channel Index
CiChaikin	Chaikin Oscillator
CiDeMarker	DeMarker
CiForce	Force Index
CiMACD	Moving Averages Convergence-Divergence
CiMomentum	Momentum
CiOsMA	Moyenne Mobile de l'Oscillateur (Histogramme du MACD)
CiRSI	Relative Strength Index
CiRVI	Relative Vigor Index
CiStochastic	Stochastic Oscillator
CiWPR	Williams' Percent Range
CiTriX	Triple Exponential Moving Averages Oscillator

© 2000-2024, MetaQuotes Ltd.

CiATR

La classe CiATR permet d'utiliser l'indicateur technique Average True Range.

Description

La classe CiATR fournit la création, la configuration et l'accès aux données de l'indicateur Average True Range.

Déclaration

```
class CiATR: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiATR

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ATR for CiATR).

© 2000-2024, MetaQuotes Ltd.

CiBearsPower

La classe CiBearsPower permet d'utiliser l'indicateur technique Bears Power (Pouvoir des Baissiers).

Description

La classe CiBearsPower fournit la création, la configuration et l'accès aux données de l'indicateur Bears Power (Puissance des Baissiers).

Déclaration

```
class CiBearsPower: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBearsPower

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_BEARS for CiBearsPower).

© 2000-2024, MetaQuotes Ltd.

CiBullsPower

La classe CiBullsPower permet d'utiliser l'indicateur technique Bulls Power (Pouvoir des Haussiers).

Description

La classe CiBullsPower fournit la création, la configuration et l'accès aux données de l'indicateur Bulls Power (Puissance des Haussiers).

Déclaration

```
class CiBullsPower: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

COBJECT
CArray
CArrayObj
CSeries
CIndicator
CiBullsPower

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_BULLS for CiBullsPower).

© 2000-2024, MetaQuotes Ltd.

CiCCI

La classe CiCCI permet d'utiliser l'indicateur technique Commodity Channel Index.

Description

La classe CiCCI fournit la création, la configuration et l'accès aux données de l'indicateur Commodity Channel Index.

Déclaration

```
class CiCCI: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiCCI

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              applied        // type de prix ou handle à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_CCI for CiCCI).

© 2000-2024, MetaQuotes Ltd.

CiChaikin

La classe CiChaikin permet d'utiliser l'indicateur technique Chaikin Oscillator.

Description

La classe CiChaikin fournit la création, la configuration et l'accès aux données de l'indicateur Chaikin Oscillator.

Déclaration

```
class CiChaikin: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

- CObject
  - CArray
    - CArrayObj
      - CSeries
        - CIndicator
          - CiChaikin

Méthodes de Classe

Attributs	
FastMaPeriod	Retourne la période de lissage de la moyenne mobile rapide
SlowMaPeriod	Retourne la période de lissage de la moyenne mobile lente
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

FastMaPeriod

Retourne la période de lissage de la moyenne mobile rapide.

```
int  FastMaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile rapide, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

SlowMaPeriod

Retourne la période de lissage de la moyenne mobile lente.

```
int  SlowMaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile lente, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string                 symbol,              // Symbole
   ENUM_TIMEFRAMES        period,              // Période
   int                    fast_ma_period,      // Période de la moyenne mobile rapide
   int                    slow_ma_period,      // Période de la moyenne mobile lente
   ENUM_MA_METHOD         ma_method,           // Méthode de lissage
   ENUM_APPLIED_VOLUME    applied              // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

fast_ma_period

[in] Période de la moyenne mobile rapide.

slow_ma_period

[in] Période de la moyenne mobile lente.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_CHAIKIN for CiChaikin).

© 2000-2024, MetaQuotes Ltd.

CiDeMarker

CiDeMarker est une classe permettant d'utiliser l'indicateur technique DeMarker.

Description

La classe CiDeMarker fournit la création, la configuration et l'accès aux données de l'indicateur DeMarker.

Déclaration

```
class CiDeMarker: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

- CObject
  - CArray
    - CArrayObj
      - CSeries
        - CIndicator
          - CiDeMarker

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_DEMARKER for CiDeMarker).

© 2000-2024, MetaQuotes Ltd.

CiForce

La classe CiForce permet d'utiliser l'indicateur technique Force Index.

Description

La classe CiForce fournit la création, la configuration et l'accès aux données de l'indicateur Force Index.

Déclaration

```
class CiForce: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiForce

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string                symbol,         // symbole
   ENUM_TIMEFRAMES       period,         // Période
   int                   ma_period,      // Période de lissage
   ENUM_MA_METHOD        ma_method,      // Méthode de lissage
   ENUM_APPLIED_VOLUME   applied         // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_FORCE for CiForce).

© 2000-2024, MetaQuotes Ltd.

CiMACD

La classe CiMACD permet d'utiliser l'indicateur technique Moving Averages Convergence-Divergence.

Description

La classe CiMACD fournit la création, la configuration et l'accès aux données de l'indicateur Moving Averages Convergence-Divergence.

Déclaration

```
class CiMACD: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

- CObject
  - CArray
    - CArrayObj
      - CSeries
        - CIndicator
          - CiMACD

Méthodes de Classe

Attributs	
FastEmaPeriod	Retourne la période de lissage de la moyenne mobile rapide.
SlowEmaPeriod	Retourne la période de lissage de la moyenne mobile lente.
SignalPeriod	Retourne la période de lissage de la ligne de signal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne principale
Signal	Retourne l'élément du buffer de la ligne de signal
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

FastEmaPeriod

Retourne la période de lissage de la moyenne mobile rapide.

```
int  FastEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile rapide, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

SlowEmaPeriod

Retourne la période de lissage de la moyenne mobile lente.

```
int  SlowEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile lente, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

SignalPeriod

Retourne la période de lissage de la ligne de signal.

```
int  SignalPeriod() const
```

Valeur de retour

Retourne la période de lissage de la ligne de signal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,              // symbole
   ENUM_TIMEFRAMES  period,              // Période
   int              fast_ema_period,     // Période de la moyenne mobile rapide
   int              slow_ema_period,     // Période de la moyenne mobile lente
   int              signal_period,       // Période du signal
   int              applied              // prix type ou handle to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

fast_ema_period

[in] Période de la moyenne mobile rapide.

slow_ema_period

[in] Période de la moyenne mobile lente.

signal_period

[in] Période de la ligne de signal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Signal

Retourne l'élément du buffer de la ligne de signal de l'index spécifié.

```
double  Signal(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne de signal à l'index spécifié, ou EMPTY_VALUE si aucune donnée correcte n'est présente.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_MACD for CiMACD).

© 2000-2024, MetaQuotes Ltd.

CiMomentum

CiMomentum est une classe permettant d'utiliser l'indicateur technique Momentum.

Description

La classe CiMomentum fournit la création, la configuration et l'accès aux données de l'indicateur Momentum.

Déclaration

```
class CiMomentum: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMomentum

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              applied        // type de prix ou handle à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_MOMENTUM for CiMomentum).

© 2000-2024, MetaQuotes Ltd.

CiOsMA

La classe CiOsMA permet d'utiliser l'indicateur technique Moving Average of Oscillator (histogramme du MACD).

Description

La classe CiOsMA fournit la création, la configuration et l'accès aux données de l'indicateur Moving Average of Oscillator (histogramme de MACD).

Déclaration

```
class CiOsMA: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiOsMA

Méthodes de Classe

Attributs	
FastEmaPeriod	Retourne la période de lissage de la moyenne mobile rapide.
SlowEmaPeriod	Retourne la période de lissage de la moyenne mobile lente.
SignalPeriod	Retourne la période de lissage de la ligne de signal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

FastEmaPeriod

Retourne la période de lissage de la moyenne mobile rapide.

```
int  FastEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile rapide, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

SlowEmaPeriod

Retourne la période de lissage de la moyenne mobile lente.

```
int  SlowEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile lente, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

SignalPeriod

Retourne la période de lissage de la ligne de signal.

```
int  SignalPeriod() const
```

Valeur de retour

Retourne la période de lissage de la ligne de signal définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,              // symbole
   ENUM_TIMEFRAMES  period,              // Période
   int              fast_ema_period,     // Période de la moyenne mobile rapide
   int              slow_ema_period,     // Période de la moyenne mobile lente
   int              signal_period,       // Période de la ligne de signal
   int              applied              // prix type ou handle to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

fast_ema_period

[in] Période de la moyenne mobile rapide.

slow_ema_period

[in] Période de la moyenne mobile lente.

signal_period

[in] Période de la ligne de signal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_OSMA for CiOsMA).

© 2000-2024, MetaQuotes Ltd.

CiRSI

La classe CiRSI permet d'utiliser l'indicateur technique Relative Strength Index.

Description

La classe CiRSI fournit la création, la configuration et l'accès aux données de l'indicateur Relative Strength Index.

Déclaration

```
class CiRSI: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiRSI

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              applied        // type de prix ou handle à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_RSI for CiRSI).

© 2000-2024, MetaQuotes Ltd.

CiRVI

La classe CiRVI permet d'utiliser l'indicateur technique Relative Vigor Index.

Description

La classe CiRVI fournit la création, la configuration et l'accès aux données de l'indicateur Relative Vigor Index.

Déclaration

```
class CiRVI: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiRVI

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne de base
Signal	Retourne l'élément du buffer de la ligne de signal
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Signal

Retourne l'élément du buffer de la ligne de signal de l'index spécifié.

```
double  Signal(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne de signal à l'index spécifié, ou EMPTY_VALUE si aucune donnée correcte n'est présente.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_RVI for CiRVI).

© 2000-2024, MetaQuotes Ltd.

CiStochastic

La classe CiChaikin permet d'utiliser l'indicateur technique Stochastique.

Description

La classe CiStochastic fournit la création, la configuration et l'accès aux données de l'indicateur Stochastique.

Déclaration

```
class CiStochastic: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiStochastic

Méthodes de Classe

Attributs	
Kperiod	Retourne la période de lissage de la ligne %K
Dperiod	Retourne la période de lissage de la ligne %D
Slowing	Retourne la période de ralentissement
MaMethod	Retourne la méthode de lissage
PriceField	Type de prix (Low/High ou Close/Close) à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne de base
Signal	Retourne l'élément du buffer de la ligne de signal
Entrée/Sortie	

© 2000-2024, MetaQuotes Ltd.

Attributs	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Kperiod

Retourne la période de lissage de la ligne %K.

```
int  Kperiod() const
```

Valeur de retour

Retourne la période de lissage de la ligne %K définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Dperiod

Retourne la période de lissage de la ligne %D.

```
int  Dperiod() const
```

Valeur de retour

Retourne la période de lissage de la ligne %D définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Slowing

Retourne la période de ralentissement.

```
int  Slowing() const
```

Valeur de retour

Retourne la période de ralentissement définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

PriceField

Retourne le prix type (Low/High ou Close/Close) to apply.

```
ENUM_STO_PRICE  PriceField() const
```

Valeur de retour

x Le prix type xprice type to apply, defined at the indicator creation.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,          // symbole
   ENUM_TIMEFRAMES  period,          // Période
   int              Kperiod,         // Période de lissage de %K
   int              Dperiod,         // Période de lissage de %D
   int              slowing,         // Période de ralentissement
   ENUM_MA_METHOD   ma_method,       // Méthode de lissage
   ENUM_STO_PRICE   price_field      // prix type to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Kperiod

[in] Période de lissage de la ligne %K.

Dperiod

[in] Période de lissage de la ligne %D.

slowing

[in] Période de ralentissement.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

price_field

[in] Prix type (Low/High ou Close/Close) to apply (ENUM_STO_PRICE enumeration).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Signal

Retourne l'élément du buffer de la ligne de signal de l'index spécifié.

```
double  Signal(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Signal

Retourne l'élément du buffer de la ligne de signal de l'index spécifié.

```
double  Signal(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne de signal à l'index spécifié, ou EMPTY_VALUE si aucune donnée correcte n'est présente.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_STOCHASTIC for CiStochastic).

© 2000-2024, MetaQuotes Ltd.

CiTriX

La classe CiTriX permet d'utiliser l'indicateur technique Triple Exponential Moving Averages Oscillator.

Description

La classe CiTriX fournit la création, la configuration et l'accès aux données de l'indicateur Triple Exponential Moving Averages Oscillator.

Déclaration

```
class CiTriX: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiTriX

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              applied        // prix type ou handle
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

applied

[in] Prix type of handle to apply.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_TRIX for CiTriX).

© 2000-2024, MetaQuotes Ltd.

CiWPR

La classe CiWPR permet d'utiliser l'indicateur technique Williams' Percent Range.

Description

La classe CiWPR fournit la création, la configuration et l'accès aux données de l'indicateur Williams' Percent Range.

Déclaration

```
class CiWPR: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Hiérarchie d'héritage

- CObject
  - CArray
    - CArrayObj
      - CSeries
        - CIndicator
          - CiWPR

Méthodes de Classe

Attributs	
CalcPeriod	Retourne la période de calcul
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

CalcPeriod

Retourne la période de calcul.

```
int  CalcPeriod() const
```

Valeur de retour

Retourne la période de calcul définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,          // symbole
   ENUM_TIMEFRAMES  period,          // Période
   int              calc_period      // Période de calcul
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

calc_period

[in] Période de calcul.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_WPR for CiWPR).

© 2000-2024, MetaQuotes Ltd.

Indicateurs de Volumes

Cet ensemble de chapitres contient les détails techniques des classes des indicateurs de Volume de la Bibliothèque Standard MQL5 et les descriptions de ses composants principaux.

Classe/groupe	Description
CiAD	Accumulation/Distribution
CiMFI	Money Flow Index
CiOBV	On Balance Volume
CiVolumes	Volumes

© 2000-2024, MetaQuotes Ltd.

CiAD

CiAD est une classe permettant d'utiliser l'indicateur technique Accumulation/Distribution.

Description

La classe CiAD fournit la création, la configuration et l'accès aux données de l'indicateur Accumulation/Distribution.

Déclaration

```
class CiAD: public CIndicator
```

Titre

```
#include <Indicators\Volumes.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiAD

Méthodes de Classe

Attributs	
Applied	Retourne le type de volume à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,      // symbole
   ENUM_TIMEFRAMES      period,      // Période
   ENUM_APPLIED_VOLUME  applied      // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_AD for CiAD).

© 2000-2024, MetaQuotes Ltd.

CiMFI

La classe CiMFI permet d'utiliser l'indicateur technique Money Flow Index.

Description

La classe CiMFI fournit la création, la configuration et l'accès aux données de l'indicateur Money Flow Index.

Déclaration

```
class CiMFI: public CIndicator
```

Titre

```
#include <Indicators\Volumes.mqh>
```

Hiérarchie d'héritage

- CObject
  - CArray
    - CArrayObj
      - CSeries
        - CIndicator
          - CiMFI

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Applied	Retourne le type de volume à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string                symbol,         // symbole
   ENUM_TIMEFRAMES       period,         // Période
   int                   ma_period,      // Période de lissage
   ENUM_APPLIED_VOLUME   applied         // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_MFI for CiMFI).

© 2000-2024, MetaQuotes Ltd.

CiOBV

La classe CiOBV permet d'utiliser l'indicateur technique On Balance Volume.

Description

La classe CiOBV fournit la création, la configuration et l'accès aux données de l'indicateur On Balance Volume.

Déclaration

```
class CiOBV: public CIndicator
```

Titre

```
#include <Indicators\Volumes.mqh>
```

Hiérarchie d'héritage

COBJECT
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiOBV

Méthodes de Classe

Attributs	
Applied	Retourne le type de volume à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,      // symbole
   ENUM_TIMEFRAMES      period,      // Période
   ENUM_APPLIED_VOLUME  applied      // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_OBV for CiOBV).

© 2000-2024, MetaQuotes Ltd.

CiVolumes

La classe CiVolumes permet d'utiliser l'indicateur technique des Volumes.

Description

La classe CiVolumes fournit la création, la configuration et l'accès aux données de l'indicateur Volumes.

Déclaration

```
class CiVolumes: public CIndicator
```

Titre

```
#include <Indicators\Volumes.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiVolumes

Méthodes de Classe

Attributs	
Applied	Retourne le type de volume à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,      // symbole
   ENUM_TIMEFRAMES      period,      // Période
   ENUM_APPLIED_VOLUME  applied      // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_VOLUMES for CiVolumes).

© 2000-2024, MetaQuotes Ltd.

Indicateurs de Bill Williams

Cet ensemble de chapitres contient les détails techniques des classes des indicateurs de Bill Williams de la Bibliothèque Standard MQL5 et les descriptions de ses composants principaux.

Classe/groupe	Description
CiAC	Oscillateur d'Accélération
CiAlligator	Alligator
CiAO	Oscillateur Awesome
CiFractals	Fractals
CiGator	Gator Oscillator
CiBWMFI	Market Facilitation Index

© 2000-2024, MetaQuotes Ltd.

CiAC

CiAC est une classe permettant d'utiliser l'oscillateur Accelerator.

Description

La classe CiAC fournit la création, la configuration et l'accès aux données de l'oscillateur Accelerator.

Déclaration

```
class CiAC: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiAC

Méthodes de Classe

Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // Période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_AC pour CiAC).

© 2000-2024, MetaQuotes Ltd.

CiAlligator

CiAlligator est une classe permettant d'utiliser l'indicateur technique Alligator.

Description

La classe CiAlligator fournit la création, la configuration et l'accès aux données de l'indicateur Alligator.

Déclaration

```
class CiAlligator: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Hiérarchie d'héritage

- CObject
  - CArray
    - CArrayObj
      - CSeries
        - CIndicator
          - CiAlligator

Méthodes de Classe

Attributs	
JawPeriod	Retourne la période de lissage pour la ligne de la Mâchoire
JawShift	Retourne le décalage horizontal de la ligne de la Mâchoire
TeethPeriod	Retourne la période de lissage de la ligne des Dents
TeethShift	Retourne le décalage horizontal de la ligne des Dents
LipsPeriod	Retourne la période de lissage de la ligne des Lévres
LipsShift	Retourne le décalage horizontal de la ligne des Lévres
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	

© 2000-2024, MetaQuotes Ltd.

Attributs	
Jaw	Retourne l'élément du buffer de la ligne de la Mâchoire
Teeth	Retourne l'élément du buffer de la ligne des Dents
Lips	Retourne l'élément du buffer de la ligne des Lévres
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

JawPeriod

Retourne la période de lissage pour la ligne de la Mâchoire.

```
int  JawPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne de la Mâchoire, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

JawShift

Retourne le décalage horizontal de la ligne de la Mâchoire.

```
int  JawShift() const
```

Valeur de retour

Décalage horizontal de la ligne de la Mâchoire, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

TeethPeriod

Retourne la période de lissage de la ligne des Dents.

```
int  TeethPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne des Dents, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

TeethShift

Retourne le décalage horizontal de la ligne des Dents.

```
int  TeethShift() const
```

Valeur de retour

Décalage horizontal de la ligne des Dents, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

LipsPeriod

Retourne la période de lissage de la ligne des Lévres.

```
int  LipsPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne des Lévres, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

LipsShift

Retourne le décalage horizontal de la ligne des Lévres.

```
int  LipsShift() const
```

Valeur de retour

Décalage horizontal de la ligne des Lévres, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,           // symbole
   ENUM_TIMEFRAMES  period,           // Période
   int              jaw_period,       // Période de la Mâchoire
   int              jaw_shift,        // Décalage de la Mâchoire
   int              teeth_period,     // Période des Dents
   int              teeth_shift,      // Décalage des Dents
   int              lips_period,      // Période des Lévres
   int              lips_shift,       // Décalage des Lévres
   ENUM_MA_METHOD   ma_method,        // Méthode de lissage
   int              applied           // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

jaw_period

[in] Période de la Mâchoire.

jaw_shift

[in] Décalage de la Mâchoire.

teeth_period

[in] Période des Dents.

teeth_shift

[in] Décalage des Dents.

lips_period

[in] Période des Lévres.

lips_shift

[in] Décalage des Lévres.

ma_method

[in] Méthode de la moyenne mobile (énumération ENUM_MA_METHOD).

applied

[in] Type de volume à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Jaw

Retourne l'élément du buffer de la ligne de la Mâchoire de l'index spécifié.

```
double  Jaw(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne de la Mâchoire de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Teeth

Retourne l'élément du buffer de la ligne des Dents de l'index spécifié.

```
double  Teeth(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément buffer de la ligne des Dents de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Lips

Retourne l'élément du buffer de la ligne des Lévres de l'index spécifié.

```
double  Lips(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément buffer de la ligne des Lévres de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ALLIGATOR for CiAlligator).

© 2000-2024, MetaQuotes Ltd.

CiAO

CiAO est une classe permettant d'utiliser l'oscillateur Awesome.

Description

La classe CiAO fournit la création, la configuration et l'accès aux données de l'oscillateur Awesome.

Déclaration

```
class CiAO: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiAO

Méthodes de Classe

Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // Période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_AO for CiAO).

© 2000-2024, MetaQuotes Ltd.

CiFractals

La classe CiFractals permet d'utiliser l'indicateur technique Fractals.

Description

La classe CiFractals fournit la création, la configuration et l'accès aux données de l'indicateur Fractals.

Déclaration

```
class CiFractals: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Hiérarchie d'héritage

COBJECT
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiFractals

Méthodes de Classe

Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Upper	Retourne l'élément du buffer supérieur
Lower	Retourne l'élément du buffer inférieur
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // Période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Upper

Retourne l'élément du buffer supérieur situé à l'index spécifié.

```
double  Upper(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer supérieur de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Lower

Retourne l'élément du buffer inférieur situé à l'index spécifié.

```
double  Lower(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer inférieur de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_FRACTALS for CiFractals).

© 2000-2024, MetaQuotes Ltd.

CiGator

La classe CiFractals permet d'utiliser l'oscillateur Gator.

Description

La classe CiGator fournit la création, la configuration et l'accès aux données de l'oscillateur Gator.

Déclaration

```
class CiGator: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiGator

Méthodes de Classe

Attributs	
JawPeriod	Retourne la période de lissage pour la ligne de la Mâchoire
JawShift	Retourne le décalage horizontal de la ligne de la Mâchoire
TeethPeriod	Retourne la période de lissage de la ligne des Dents
TeethShift	Retourne le décalage horizontal de la ligne des Dents
LipsPeriod	Retourne la période de lissage de la ligne des Lévres
LipsShift	Retourne le décalage horizontal de la ligne des Lévres
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Upper	Retourne l'élément du buffer supérieur

© 2000-2024, MetaQuotes Ltd.

Attributs	
Lower	Retourne l'élément du buffer inférieur
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

JawPeriod

Retourne la période de lissage pour la ligne de la Mâchoire.

```
int  JawPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne de la Mâchoire, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

JawShift

Retourne le décalage horizontal de la ligne de la Mâchoire.

```
int  JawShift() const
```

Valeur de retour

Décalage horizontal de la ligne de la Mâchoire, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

TeethPeriod

Retourne la période de lissage de la ligne des Dents.

```
int  TeethPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne des Dents, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

TeethShift

Retourne le décalage horizontal de la ligne des Dents.

```
int  TeethShift() const
```

Valeur de retour

Décalage horizontal de la ligne des Dents, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

LipsPeriod

Retourne la période de lissage de la ligne des Lévres.

```
int  LipsPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne des Lévres, définie à la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

LipsShift

Retourne le décalage horizontal de la ligne des Lévres.

```
int  LipsShift() const
```

Valeur de retour

Décalage horizontal de la ligne des Lévres, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,           // symbole
   ENUM_TIMEFRAMES  period,           // Période
   int              jaw_period,       // Période de la Mâchoire
   int              jaw_shift,        // Décalage de la Mâchoire
   int              teeth_period,     // Période des Dents
   int              teeth_shift,      // Décalage des Dents
   int              lips_period,      // Période des Lévres
   int              lips_shift,       // Décalage des Lévres
   ENUM_MA_METHOD   ma_method,        // Méthode de lissage
   int              applied           // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

jaw_period

[in] Période de la Mâchoire.

jaw_shift

[in] Décalage de la Mâchoire.

teeth_period

[in] Période des Dents.

teeth_shift

[in] Décalage des Dents.

lips_period

[in] Période des Lévres.

lips_shift

[in] Décalage des Lévres.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Type de volume à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Upper

Retourne l'élément du buffer supérieur situé à l'index spécifié.

```
double  Upper(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer supérieur de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Lower

Retourne l'élément du buffer inférieur situé à l'index spécifié.

```
double  Upper(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer inférieur de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_GATOR for CiGator).

© 2000-2024, MetaQuotes Ltd.

CiBWMFI

CiBWMFI est une classe permettant d'utiliser l'indicateur technique Market Facilitation Index de Bill Williams.

Description

La classe CiBWMFI fournit la création, la configuration et l'accès aux données de l'indicateur Market Facilitation Index de Bill Williams.

Déclaration

```
class CiBWMFI: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Hiérarchie d'héritage

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBWMFI

Méthodes de Classe

Attributs	
Applied	Retourne le type de volume à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Méthodes héritées de la classe CArray

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Méthodes héritées de la classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Méthodes héritées de la classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Méthodes héritées de la classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,      // symbole
   ENUM_TIMEFRAMES      period,      // Période
   ENUM_APPLIED_VOLUME  applied      // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2024, MetaQuotes Ltd.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index       // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_BWMFI for CiBWMFI).

© 2000-2024, MetaQuotes Ltd.

CiCustom

CiCustom est une classe prévue pour utiliser les indicateurs techniques personnalisés.

Description

La classe CiCustom fournit la création, la configuration et l'accès aux données de l'indicateur personnalisés.

Déclaration

```
class CiCustom: public CIndicator
```

Titre

```
#include <Indicators\Custom.mqh>
```

Méthodes de Classe

Attributs	
NumBuffers	Définit le nombre de buffers
NumParams	Retourne le nombre de paramètres
ParamType	Retourne le type du paramètre spécifié
ParamLong	Retourne la valeur du paramètre spécifié de type integer
ParamDouble	Retourne la valeur du paramètre spécifié de type double
ParamString	Retourne la valeur du paramètre spécifié de type string
Entrée/Sortie	
virtual Type	REtourne l'identifiant du type de l'objet

© 2000-2024, MetaQuotes Ltd.

NumBuffers

Définit le nombre de buffers.

```
bool  NumBuffers(
   int  buffers      // nombre de buffers
   )
```

Valeur de retour

vrai si réalisé avec succès, faux si les buffers n'ont pas été définis.

© 2000-2024, MetaQuotes Ltd.

NumParams

Retourne le nombre de paramètres

```
int  NumParams() const
```

Valeur de retour

Nombre de paramètres utilisés pour créer l'indicateur.

© 2000-2024, MetaQuotes Ltd.

ParamType

Retourne le type du paramètre à l'index spécifié.

```
ENUM_DATATYPE  ParamType(
   int  index      // index du paramètre
   ) const
```

Paramètres

index

[in] Index du paramètre.

Valeur de retour

Retourne le type de donnée (valeur de l'énumération ENUM_DATATYPE) du paramètre à l'index spécifié, utilisé pour créer l'indicateur.

Note

Si l'index du paramètre est invalide, retourne WRONG_VALUE.

© 2000-2024, MetaQuotes Ltd.

ParamLong

Retourne la valeur du paramètre spécifié de type long.

```
long  ParamLong(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index du paramètre.

Valeur de retour

La valeur du paramètre spécifié de type long utilisé pour créer l'indicateur.

Note

Si le nombre est invalide, retourne 0.

© 2000-2024, MetaQuotes Ltd.

ParamDouble

Retourne la valeur du paramètre spécifié de type double.

```
double  ParamDouble(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index du paramètre.

Valeur de retour

La valeur du paramètre spécifié de type double utilisé pour créer l'indicateur.

Note

Si le nombre est invalide, retourne EMPTY_VALUE.

© 2000-2024, MetaQuotes Ltd.

ParamString

Retourne la valeur du paramètre spécifié de type string.

```
string  ParamString(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index du paramètre.

Valeur de retour

La valeur du paramètre spécifié de type string utilisé pour créer l'indicateur.

Note

Si le nombre est invalide, retourne une chaîne vide.

© 2000-2024, MetaQuotes Ltd.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_CUSTOM for CiCustom).

© 2000-2024, MetaQuotes Ltd.

Classes de Trade

Cette section contient les détails techniques d'utilisation des classes de trade et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création des programmes personnalisés (Expert Advisors).

La Bibliothèque Standard MQL5 (en terme d'ensemble de données) est située dans le répertoire de travail du terminal dans le répertoire Include\Arrays.

Classe/Groupe	Description
CAccountInfo	Classe pour travailler avec les propriétés du compte de trading
CSymbolInfo	Classe pour travailler avec les propriétés des instruments de trading
COrderInfo	Classe pour travailler avec les propriétés des ordres en attente
CHistoryOrderInfo	Classe pour travailler avec les propriétés de l'historique des ordres
CPositionInfo	Classe pour travailler avec les propriétés des positions ouvertes
CDealInfo	Classe pour travailler avec les propriétés de l'historique des deals
CTrade	Classe pour l'exécution des opérations de trading
CTerminalInfo	Classe pour récupérer les propriétés de l'environnement du programme MQL5

© 2000-2024, MetaQuotes Ltd.

CAccountInfo

La classe CAccountInfo permet d'accéder aux propriétés du compte actuellement utilisé.

Description

La classe CAccountInfo permet d'accéder aux propriétés du compte actuellement utilisé.

Déclaration

```
class CAccountInfo : public CObject
```

Titre

```
#include <Trade\AccountInfo.mqh>
```

Hiérarchie d'héritage

CObject
  CAccountInfo

Méthodes de classe par groupes

Accès aux propriétés de type integer	
Login	Retourne le numéro du compte
TradeMode	Retourne le mode de trading
TradeModeDescription	Retourne le mode de trading sous forme d'une chaîne de caractères
Leverage	Retourne le levier actuel
StopoutMode	Retourne le mode de stop out du compte
StopoutModeDescription	Retourne la description du mode de stop out du compte
TradeAllowed	Retourne si les opérations de trading sont autorisées ou pas
TradeExpert	Retourne si les Expert Advisors sont autorisés à effectuer des opérations de trading ou pas
LimitOrders	Retourne le nombre maximal d'ordres en attente autorisés
MarginMode	Retourne le mode de calcul de la marge
MarginModeDescription	Retourne le mode de calcul de la marge sous forme d'une chaîne de caractères
Accès aux propriétés de type double	
Balance	Retourne la balance du compte

© 2000-2024, MetaQuotes Ltd.

Accès aux propriétés de type integer	
Credit	Retourne le montant du crédit
Profit	Retourne le montant du profit actuel du compte
Equity	Retourne le montant actuel du compte
Margin	Retourne le montant de la marge réservée
FreeMargin	Retourne le montant de la marge libre
MarginLevel	Retourne le montant de la marge
MarginCall	Retourne le montant de la marge de dépôt
MarginStopOut	Retourne le niveau de marge de Stop Out
Accès aux propriétés de type texte	
Name	Retourne le nom du client
Server	Retourne le nom du serveur de trading
Currency	Retourne le nom de la devise actuelle du compte
Company	Retourne le nom de la société qui fournit le compte
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string
Méthodes supplémentaires	
OrderProfitCheck	Retourne le profit évalué, basé sur les paramètres passés
MarginCheck	Retourne le montant de la marge requise pour exécuter une opération de trading
FreeMarginCheck	Retourne le montant de la marge libre restante après exécution d'une opération de trading
MaxLotCheck	Retourne le volume maximal possible d'une opération de trading

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Login

Retourne le numéro du compte.

```
long  Login() const
```

Valeur de retour

Numéro du compte.

© 2000-2024, MetaQuotes Ltd.

TradeMode

Retourne le mode de trading.

```
ENUM_ACCOUNT_TRADE_MODE  TradeMode() const
```

Valeur de retour

Mode de trading (valeur de l'énumération ENUM_ACCOUNT_TRADE_MODE).

© 2000-2024, MetaQuotes Ltd.

TradeModeDescription

Retourne le mode de trading sous forme d'une chaîne de caractères.

```
string  TradeModeDescription() const
```

Valeur de retour

Mode de trading sous forme d'une chaîne de caractères.

© 2000-2024, MetaQuotes Ltd.

Leverage

Retourne le niveau du levier actuel.

```
long  Leverage() const
```

Valeur de retour

Niveau de levier actuel.

© 2000-2024, MetaQuotes Ltd.

StopoutMode

Retourne le mode de Stop Out du compte (stop automatique en cas de perte totale du montant du compte).

```
ENUM_ACCOUNT_STOPOUT_MODE  StopoutMode() const
```

Valeur de retour

Mode de Stop Out du compte (valeur de l'énumération ENUM_ACCOUNT_STOPOUT_MODE).

© 2000-2024, MetaQuotes Ltd.

StopoutModeDescription

Retourne le mode de définition du niveau de marge minimum sous forme d'une chaîne de caractères.

```
string  StopoutModeDescription() const
```

Valeur de retour

Le mode de définition du niveau de marge minimum sous forme d'une chaîne de caractère.

© 2000-2024, MetaQuotes Ltd.

MarginMode

Retourne le mode de calcul de la marge

```
ENUM_ACCOUNT_MARGIN_MODE  MarginMode() const
```

Valeur de Retour

Le mode de calcul de la marge comme une valeur de l'énumération ENUM_ACCOUNT_MARGIN_MODE.

© 2000-2024, MetaQuotes Ltd.

MarginModeDescription

Retourne le mode de calcul de la marge sous forme d'une chaîne de caractères

```
string  MarginModeDescription() const
```

Valeur de Retour

Mode de calcul de la marge sous forme d'une chaîne de caractères.

© 2000-2024, MetaQuotes Ltd.

TradeAllowed

Retourne le flag de trade allowance

```
bool  TradeAllowed() const
```

Valeur de retour

Flag de trade allowance

© 2000-2024, MetaQuotes Ltd.

TradeExpert

Retourne le flag de trade allowance automatique.

```
bool TradeExpert() const
```

Valeur de retour

Flag de trade allowance automatique.

© 2000-2024, MetaQuotes Ltd.

LimitOrders

Retourne le nombre maximal d'ordres en attente autorisés

```
int  LimitOrders() const
```

Valeur de retour

Le nombre maximal d'ordres en attente autorisés.

Note

0 - aucune limite.

© 2000-2024, MetaQuotes Ltd.

Balance

Retourne la balance du compte.

```
double  Balance() const
```

Valeur de retour

La balance du compte (dans la devise du compte).

© 2000-2024, MetaQuotes Ltd.

Credit

Retourne le montant du crédit.

```
double  Credit() const
```

Valeur de retour

Montant du crédit (dans la devise du compte).

© 2000-2024, MetaQuotes Ltd.

Profit

Retourne le montant du profit actuel du compte.

```
double  Profit() const
```

Valeur de retour

Montant actuel du profit compte (dans la devise du compte).

© 2000-2024, MetaQuotes Ltd.

Equity

Retourne le montant actuel du compte.

```
double  Equity() const
```

Valeur de retour

Montant actuel du compte (dans la devise du compte).

© 2000-2024, MetaQuotes Ltd.

Margin

Retourne le montant de la marge réservée.

```
double  Margin() const
```

Valeur de retour

Montant de la marge réservée (dans la devise du compte).

© 2000-2024, MetaQuotes Ltd.

FreeMargin

Retourne le montant de la marge libre.

```
double  FreeMargin() const
```

Valeur de retour

Montant de la marge libre (dans la devise du compte).

© 2000-2024, MetaQuotes Ltd.

MarginLevel

Retourne le niveau de marge.

```
double  MarginLevel() const
```

Valeur de retour

Niveau de marge.

© 2000-2024, MetaQuotes Ltd.

MarginCall

Retourne le montant de la marge de dépôt.

```
double  MarginCall() const
```

Valeur de retour

Montant de la marge de dépôt.

© 2000-2024, MetaQuotes Ltd.

MarginStopOut

Retourne le niveau de marge de Stop Out.

```
double  MarginStopOut() const
```

Valeur de retour

Le niveau de marge de Stop Out.

© 2000-2024, MetaQuotes Ltd.

Name

Retourne le nom du client.

```
string  Name() const
```

Valeur de retour

Nom du client.

© 2000-2024, MetaQuotes Ltd.

Server

Retourne le nom du serveur de trading.

```
string  Server() const
```

Valeur de retour

Nom du serveur de trading.

© 2000-2024, MetaQuotes Ltd.

Currency

Retourne le nom de devise actuelle.

```
string  Currency() const
```

Valeur de retour

Nom de la devise du compte.

© 2000-2024, MetaQuotes Ltd.

Company

Retourne le nom de la société qui fournit le compte.

```
string  Company() const
```

Valeur de retour

Nom de la société fournissant le compte.

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
long  InfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  prop_id     // Identifiant de la propriété
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété. La valeur peut être une des valeurs de l'énumération ENUM_ACCOUNT_INFO_INTEGER .

Valeur de retour

Valeur de type long.

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
double  InfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  prop_id     // Identifiant de la propriété
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété. La valeur peut être une des valeurs de l'énumération ENUM_ACCOUNT_INFO_DOUBLE .

Valeur de retour

Valeur de type double.

© 2000-2024, MetaQuotes Ltd.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
string  InfoString(
   ENUM_ACCOUNT_INFO_STRING  prop_id     // Identifiant de la propriété
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété. La valeur peut être une des valeurs de l'énumération ENUM_ACCOUNT_INFO_STRING .

Valeur de retour

Valeur de type string.

© 2000-2024, MetaQuotes Ltd.

OrderProfitCheck

La fonction calcule le profit du compte courant basé sur les paramètres passés. La fonction est utilisée pour pré-évaluer le résultat d'une opération de trading. La valeur est retournée dans la devise du compte.

```
double  OrderProfitCheck(
   const string        symbol,              // symbole
   ENUM_ORDER_TYPE     trade_operation,     // opération type (ORDER_TYPE_BUY or ORDE
   double              volume,              // volume
   double              price_open,          // prix d'ouverture
   double              price_close          // prix de clôture
   ) const
```

Paramètres

symbol

[in] Symbole de l'opération de trading.

trade_operation

[in] Type de l'opération de trading (énumération ***ENUM_ORDER_TYPE).volume[in] Volume de l'opération de trading.price_open[in] Prix d'ouverture.price_close[in] Prix de clôture.Valeur de retourEn cas de succès, retourne le montant du profit, ou EMPTY_VALUE en cas d'erreur.

© 2000-2024, MetaQuotes Ltd.

MarginCheck

Retourne le montant de la marge requise pour exécuter une opération de trading.

```
double  MarginCheck(
   const string        symbol,              // symbole
   ENUM_ORDER_TYPE     trade_operation,     // opération
   double              volume,              // volume
   double              price                // prix
   ) const
```

Paramètres

symbol

[in] Symbole de l'opération de trading.

trade_operation

[in] Type de l'opération de trading (énumération ***ENUM_ORDER_TYPE).volume[in] Volume de l'opération de trading.price[in] Prix of trade operation.Valeur de retourLe montant de la marge requise pour exécuter une opération de trading.

© 2000-2024, MetaQuotes Ltd.

FreeMarginCheck

Retourne le montant de la marge libre restante après une opération de trading.

```
double  FreeMarginCheck(
   const string        symbol,              // symbole
   ENUM_ORDER_TYPE     trade_operation,     // opération
   double              volume,              // volume
   double              price                // prix
   ) const
```

Paramètres

symbol

[in] Symbole de l'opération de trading.

trade_operation

[in] Type de l'opération de trading (énumération ***ENUM_ORDER_TYPE).volume[in] Volume de l'opération de trading.price[in] Prix of trade operation.Valeur de retourMontant de la marge libre restante après une opération de trading.

© 2000-2024, MetaQuotes Ltd.

MaxLotCheck

Gets the maximum possible Volume de l'opération de trading.

```
double  MaxLotCheck(
   const string        symbol,              // symbole
   ENUM_ORDER_TYPE     trade_operation,     // opération
   double              price,               // prix
   double              percent=100          // pourcentage de la marge disponible (0-1
   ) const
```

Paramètres

symbol

[in] Symbole de l'opération de trading.

trade_operation

[in] Type de l'opération de trading (énumération ***ENUM_ORDER_TYPE).price[in] Prix of trade operation.percent=100[in] Pourcentage de la marge disponible (0-100%), utilisé pour l'opération de trading.Valeur de retourMaximum possible Volume de l'opération de trading.

© 2000-2024, MetaQuotes Ltd.

CSymbolInfo

La classe CSymbolInfo permet d'accéder aux propriétés du symbole.

Description

La classe CSymbolInfo fournit l'accès aux propriétés du symbole.

Déclaration

```
class CSymbolInfo : public CObject
```

Titre

```
#include <Trade\SymbolInfo.mqh>
```

Hiérarchie d'héritage

CObject
  CSymbolInfo

Méthodes de classe par groupes

Contrôle	
Refresh	Rafraîchit les données du symbole
RefreshRates	Rafraîchit les cotations du symbole
Propriétés	
Name	Retourne/Définit le nom du symbole
Select	Retourne/Définit le flag "Market Watch" du symbole
IsSynchronized	Vérifie la synchronisation du symbole avec le serveur
Volumes	
Volume	Retourne le volume du dernier deal
VolumeHigh	Retourne le volume maximal d'une journée
VolumeLow	Retourne le volume minimal d'une journée
Divers	
Time	Retourne l'heure de la dernière cotation
Spread	Retourne le montant du spread (en points)
SpreadFloat	Retourne le flag de spread flottant
TicksBookDepth	Retourne la profondeur des ticks
Niveaux	

© 2000-2024, MetaQuotes Ltd.

Contrôle	
StopsLevel	Retourne l'incrément minimal pour les ordres (en points)
FreezeLevel	Retourne la distance de gel des opérations de trading (en points)
Prix de l'Offre (Bid)	
Bid	Retourne le prix de l'Offre (Bid) courant
BidHigh	Retourne le prix Bid maximal d'une journée
BidLow	Retourne le prix Bid minimal d'une journée
Prix de la Demande (Ask)	
Ask	Retourne le prix de la Demande (Ask) courant
AskHigh	Retourne le prix Ask maximal d'une journée
AskLow	Retourne le prix Ask minimal d'une journée
Prix	
Last	Retourne le prix Last courant
LastHigh	Retourne le prix Last maximal d'une journée
LastLow	Retourne le prix Last minimal d'une journée
Modes de trading	
TradeCalcMode	Retourne le mode de calcul du coût d'un contrat
TradeCalcModeDescription	Retourne le mode de calcul du coût d'un contrat sous forme d'une chaîne de caractères
TradeMode	Retourne le type d'exécution d'un ordre
TradeModeDescription	Retourne le type d'exécution de l'ordre sous forme d'une chaîne de caractères
TradeExecution	Retourne le mode de clôture des deals
TradeExecutionDescription	Retourne le mode de clôture des deals sous forme d'une chaîne de caractères
Swaps	
SwapMode	Retourne le modèle de calcul du swap
SwapModeDescription	Retourne le modèle de calcul du swap sous forme d'une chaîne de caractères
SwapRollover3days	Retourne le coût de 3 jours de swap
SwapRollover3daysDescription	Retourne le coût de 3 jours de swap sous forme d'une chaîne de caractères
Marge et flags	

© 2000-2024, MetaQuotes Ltd.

Contrôle	
MarginInitial	Retourne la valeur de la marge initiale
MarginMaintenance	Retourne la valeur de la marge de maintenance
MarginLong	Retourne le taux de marge appliqué pour les positions longues
MarginShort	Retourne le taux de marge appliqué pour les positions courtes
MarginLimit	Retourne le taux de marge appliqué pour les ordres Limit
MarginStop	Retourne le taux de marge appliqué pour les ordres Stop
MarginStopLimit	Retourne le taux de marge appliqué pour les ordres StopLimit
TradeTimeFlags	Retourne les flags des modes autorisés d'expiration d'un ordre
TradeFillFlags	Retourne les flags des modes autorisés de remplissage d'un ordre
Quantification	
Digits	Retourne le nombre de décimales (nombre de chiffres après la virgule)
Point	Retourne la valeur d'un point
TickValue	Retourne le coût d'un tick (changement minimal du prix)
TickValueProfit	Retourne le prix calculé du tick pour une position gagnante
TickValueLoss	Retourne le prix calculé du tick pour une position perdante
TickSize	Retourne le changement minimal du prix
Tailles des contrats	
ContractSize	Retourne le montant du contrat de trading
LotsMin	Retourne le volume minimal pour clôturer un deal
LotsMax	Retourne le volume maximal pour clôturer un deal
LotsStep	Retourne le pas minimal de changement de volume pour clôturer un deal
LotsLimit	Retourne le volume maximal autorisé pour une position ouverte et pour les ordres en attente (sans distinction de la direction) pour un symbole
Tailles de swaps	
SwapLong	Retourne la valeur de swap d'une position longue
SwapShort	Retourne la valeur de swap d'une position courte
Propriétés texte	
CurrencyBase	Retourne le nom de la devise de base du symbole
CurrencyProfit	Retourne le nom de la devise de profit

© 2000-2024, MetaQuotes Ltd.

Contrôle	
CurrencyMargin	Retourne le nom de la devise de marge
Bank	Retourne le nom de la source de la cotation actuelle
Description	Retourne la description du symbole sous forme d'une chaîne de caractères
Path	Retourne le chemin dans l'arbre des symboles
Propriétés du symbole	
SessionDeals	Retourne le nombre de deals de la session courante
SessionBuyOrders	Retourne le nombre d'ordres Buy (ordres d'achat) au moment de l'appel
SessionSellOrders	Retourne le nombre d'ordres Sell (ordres de vente) au moment de l'appel
SessionTurnover	Retourne le résumé du turnover de la session courante
SessionInterest	Retourne le résumé de l'open interest de la session courante
SessionBuyOrdersVolume	Retourne le volume courant des ordres Buy (ordres d'achat)
SessionSellOrdersVolume	Retourne le volume courant des ordres Sell (ordres de vente)
SessionOpen	Retourne le prix d'ouverture de la session courante
SessionClose	Retourne le prix de clôture de la session courante
SessionAW	Retourne le prix moyen pondéré de la session courante
SessionPriceSettlement	Retourne le prix de compensation (settlement price) de la session courante
SessionPriceLimitMin	Retourne le prix minimal de la session courante
SessionPriceLimitMax	Retourne le prix maximal de la session courante
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string.
Fonctions d'aide	
NormalizePrice	Retourne la valeur du prix, normalisée en utilisant les propriétés du symbole

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Refresh

Rafraîchit les données du symbole.

```
void  Refresh()
```

Valeur de retour

Aucune.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

RefreshRates

Rafraîchit les cotations du symbole.

```
bool  RefreshRates()
```

Valeur de retour

vrai en cas de succès, faux si les cotations ne peuvent pas être rafraîchies.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Name

Retourne le nom du symbole.

```
string  Name() const
```

Valeur de retour

nom du symbole.

Name

Définit le nom du symbole.

```
bool  Name(string name)
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Select

Retourne le flag "Market Watch" du symbole.

```
bool  Select() const
```

Valeur de retour

Le flag "Market Watch" du symbole.

Select

Définit le flag "Market Watch" du symbole.

```
bool  Select()
```

Valeur de retour

vrai en cas de succès, faux si le flag n'a pas pu être modifié.

© 2000-2024, MetaQuotes Ltd.

IsSynchronized

Vérifie la synchronisation du symbole avec le serveur.

```
bool  IsSynchronized() const
```

Valeur de retour

vrai si le symbole est synchronisé avec le serveur, sinon faux.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Volume

Retourne le volume du dernier deal.

```
long  Volume() const
```

Valeur de retour

Le volume du dernier deal.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

VolumeHigh

Retourne le volume maximal d'une journée.

```
long  VolumeHigh() const
```

Valeur de retour

Le volume maximal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

VolumeLow

Retourne le volume minimal d'une journée.

```
long  VolumeLow() const
```

Valeur de retour

Le volume minimal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Time

Retourne l'heure de la dernière cotation.

```
datetime  Time() const
```

Valeur de retour

L'heure de la dernière cotation.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Spread

Retourne le montant du spread (en points).

```
int  Spread() const
```

Valeur de retour

Le montant du spread (en points).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SpreadFloat

Retourne le flag de spread flottant.

```
bool  SpreadFloat() const
```

Valeur de retour

Le flag de spread flottant.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TicksBookDepth

Retourne la profondeur de sauvegarde des ticks.

```
int  TicksBookDepth() const
```

Valeur de retour

La profondeur de sauvegarde des ticks.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

StopsLevel

Retourne le niveau minimal du stop pour les ordres (en points).

```
int  StopsLevel() const
```

Valeur de retour

Le niveau minimal du stop pour les ordres (en points).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

FreezeLevel

Retourne le montant de gel (en points)

```
int  FreezeLevel() const
```

Valeur de retour

Distance du niveau de gel (en points).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Bid

Retourne le prix de l'Offre (Bid) courant.

```
double  Bid() const
```

Valeur de retour

Le prix de l'Offre (Bid) courant.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

BidHigh

Retourne le prix Bid maximal d'une journée.

```
double  BidHigh() const
```

Valeur de retour

Le prix Bid maximal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

BidLow

Retourne le prix Bid minimal d'une journée.

```
double  BidLow() const
```

Valeur de retour

Le prix Bid minimal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Ask

Retourne le prix de la Demande (Ask) courant.

```
double  Ask() const
```

Valeur de retour

Prix de la Demande (Ask)

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

AskHigh

Retourne le prix Ask maximal d'une journée.

```
double  AskHigh() const
```

Valeur de retour

Le prix Ask maximal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

AskLow

Retourne le prix Ask minimal d'une journée.

```
double  AskLow() const
```

Valeur de retour

Le prix Ask minimal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Last

Retourne le prix Last courant.

```
double  Last() const
```

Valeur de retour

Prix Last courant.

© 2000-2024, MetaQuotes Ltd.

LastHigh

Retourne le prix Last maximal d'une journée.

```
double  LastHigh() const
```

Valeur de retour

Le prix Last maximal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

LastLow

Retourne le prix Last minimal d'une journée.

```
double  LastLow() const
```

Valeur de retour

Le prix Last minimal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TradeCalcMode

Retourne le mode de calcul du coût d'un contrat.

```
ENUM_SYMBOL_CALC_MODE  TradeCalcMode() const
```

Valeur de retour

Le mode de calcul du coût d'un contrat (valeur de l'énumération ENUM_SYMBOL_CALC_MODE).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TradeCalcModeDescription

Retourne le mode de calcul du coût d'un contrat sous forme d'une chaîne de caractères.

```
string  TradeCalcModeDescription() const
```

Valeur de retour

Le mode de calcul du coût d'un contrat sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TradeMode

Retourne le type d'exécution d'un ordre.

```
ENUM_SYMBOL_TRADE_MODE  TradeMode() const
```

Valeur de retour

Le type d'exécution d'un ordre (valeur de l'énumération ENUM_SYMBOL_TRADE_MODE).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TradeModeDescription

Retourne le mode de trading sous forme d'une chaîne de caractères.

```
string  TradeModeDescription() const
```

Valeur de retour

Mode de trading sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TradeExecution

Retourne le mode d'exécution d'un trade.

```
ENUM_SYMBOL_TRADE_EXECUTION  TradeExecution() const
```

Valeur de retour

Le mode d'exécution d'un trade (valeur de l'énumération ENUM_SYMBOL_TRADE_EXECUTION).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TradeExecutionDescription

Retourne le mode d'exécution d'un trade sous forme d'une chaîne de caractères.

```
string  TradeExecutionDescription() const
```

Valeur de retour

Le mode d'exécution d'un trade sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SwapMode

Retourne le modèle de calcul du swap.

```
ENUM_SYMBOL_SWAP_MODE  SwapMode() const
```

Valeur de retour

Le modèle de calcul du swap (valeur de l'énumération ENUM_SYMBOL_SWAP_MODE).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SwapModeDescription

Retourne le modèle de calcul du swap sous forme d'une chaîne de caractères.

```
string  SwapModeDescription() const
```

Valeur de retour

Le modèle de calcul du swap sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SwapRollover3days

Retourne le jour du retournement du swap.

```
ENUM_DAY_OF_WEEK  SwapRollover3days() const
```

Valeur de retour

Le jour de retournement du swap (valeur de l'énumération ENUM_DAY_OF_WEEK).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SwapRollover3daysDescription

Retourne le jour du retournement du swap sous forme d'une chaîne de caractères.

```
string  SwapRollover3daysDescription() const
```

Valeur de retour

Le jour du retournement du swap sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

MarginInitial

Retourne la valeur de la marge initiale.

```
double  MarginInitial()
```

Valeur de retour

La valeur de la marge initiale.

Note

Correspond au montant de la marge (dans la devise de marge de l'instrument) qui est chargé pour un lot. Utilisé pour vérifier le capital du client lorsqu'il entre sur le marché.

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

MarginMaintenance

Retourne la valeur de la marge de maintenance.

```
double  MarginMaintenance()
```

Valeur de retour

La valeur de la marge de maintenance.

Note

Correspond au montant de la marge (dans la devise de marge de l'instrument) qui est chargé pour un lot. Utilisé pour vérifier le capital du client lorsque l'état du compte est modifié. Si la marge de maintenance est égale à 0, la marge initiale est alors utilisée.

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

MarginLong

Retourne le taux de marge appliqué pour les positions longues.

```
double  MarginLong() const
```

Valeur de retour

Le taux de marge appliqué pour les positions longues.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

MarginShort

Retourne le taux de marge appliqué pour les positions courtes.

```
double  MarginShort() const
```

Valeur de retour

Le taux de marge appliqué pour les positions courtes.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

MarginLimit

Retourne le taux de marge appliqué aux ordres Limit.

```
double  MarginLimit() const
```

Valeur de retour

Le taux de marge appliqué aux ordres Limit.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

MarginStop

Retourne le taux de marge appliqué aux ordres Stop.

```
double  MarginStop() const
```

Valeur de retour

Le taux de marge appliqué aux ordres Stop.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

MarginStopLimit

Retourne le taux de marge appliqué aux ordres Stop Limit.

```
double  MarginStopLimit() const
```

Valeur de retour

Le taux de marge appliqué aux ordres Stop Limit.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TradeTimeFlags

Retourne les flags des modes autorisés d'expiration d'un ordre.

```
int  TradeTimeFlags() const
```

Valeur de retour

Les flags des modes autorisés d'expiration d'un ordre.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TradeFillFlags

Retourne les flags des modes autorisés de remplissage d'un ordre.

```
int  TradeFillFlags() const
```

Valeur de retour

Les flags des modes autorisés de remplissage d'un ordre.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Digits

Retourne le nombre de décimales (nombre de chiffres après la virgule).

```
int  Digits() const
```

Valeur de retour

Retourne le nombre de décimales (nombre de chiffres après la virgule).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Point

Retourne la valeur d'un point.

```
double  Point() const
```

Valeur de retour

Valeur d'un point.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TickValue

Retourne le coût d'un tick (changement minimal du prix).

```
double  TickValue() const
```

Valeur de retour

Le coût d'un tick (changement minimal du prix).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TickValueProfit

Retourne le prix calculé du tick pour une position gagnante.

```
double  TickValueProfit() const
```

Valeur de retour

Le prix calculé du tick pour une position gagnante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TickValueLoss

Retourne le prix calculé du tick pour une position perdante.

```
double  TickValueLoss() const
```

Valeur de retour

Le prix calculé du tick pour une position perdante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

TickSize

Retourne le changement minimal du prix.

```
double  TickSize() const
```

Valeur de retour

Le changement minimal du prix.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

ContractSize

Retourne le montant du contrat de trading.

```
double  ContractSize() const
```

Valeur de retour

Le montant du contrat de trading.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

LotsMin

Retourne le volume minimal pour clôturer un deal.

```
double  LotsMin() const
```

Valeur de retour

Le volume minimal pour clôturer un deal.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

LotsMax

Retourne le volume maximal pour clôturer un deal.

```
double  LotsMax() const
```

Valeur de retour

Le volume maximal pour clôturer un deal.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

LotsStep

Retourne le pas minimal de changement de volume pour clôturer un deal.

```
double  LotsStep() const
```

Valeur de retour

Le pas minimal de changement de volume pour clôturer un deal.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

LotsLimit

Retourne le volume maximal autorisé pour une position ouverte et pour les ordres en attente (sans distinction de la direction) pour un symbole.

```
double  LotsLimit() const
```

Valeur de retour

Le volume maximal autorisé pour une position ouverte et pour les ordres en attente (sans distinction de la direction) pour un symbole.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SwapLong

Retourne la valeur de swap d'une position longue.

```
double  SwapLong() const
```

Valeur de retour

La valeur de swap d'une position longue.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SwapShort

Retourne la valeur de swap d'une position courte.

```
double  SwapShort() const
```

Valeur de retour

La valeur de swap d'une position courte.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

CurrencyBase

Retourne le nom de la devise de base du symbole.

```
string  CurrencyBase() const
```

Valeur de retour

Le nom de la devise de base du symbole.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

CurrencyProfit

Retourne le nom de la devise de profit.

```
string  CurrencyProfit() const
```

Valeur de retour

Le nom de la devise de profit.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

CurrencyMargin

Retourne le nom de la devise de marge.

```
string  CurrencyMargin() const
```

Valeur de retour

Le nom de la devise de marge.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Bank

Retourne le nom de la source de la cotation actuelle.

```
string  Bank() const
```

Valeur de retour

Le nom de la source de la cotation actuelle.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Description

Retourne la description du symbole sous forme d'une chaîne de caractères.

```
string  Description() const
```

Valeur de retour

La description du symbole sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

Path

Le chemin dans l'arbre des symboles.

```
string  Path() const
```

Valeur de retour

Le chemin dans l'arbre des symboles.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionDeals

Retourne le nombre de deals de la session courante.

```
long  SessionDeals() const
```

Valeur de retour

Retourne le nombre de deals de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionBuyOrders

Retourne le nombre d'ordres Buy (ordres d'achat) au moment de l'appel.

```
long  SessionBuyOrders() const
```

Valeur de retour

Le nombre d'ordres Buy (ordres d'achat) au moment de l'appel.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionSellOrders

Retourne le nombre d'ordres Sell (ordres de vente) au moment de l'appel.

```
long  SessionSellOrders() const
```

Valeur de retour

Le nombre d'ordres Sell (ordres de vente) au moment de l'appel.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionTurnover

Retourne le résumé du turnover de la session courante.

```
double  SessionTurnover() const
```

Valeur de retour

Le résumé du turnover de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionInterest

Retourne le résumé de l'open interest de la session courante.

```
double  SessionInterest() const
```

Valeur de retour

Le résumé de l'open interest de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionBuyOrdersVolume

Retourne le volume courant des ordres Buy (ordres d'achat).

```
double  SessionBuyOrdersVolume() const
```

Valeur de retour

Le volume courant des ordres Buy (ordres d'achat).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionSellOrdersVolume

Retourne le volume courant des ordres Sell (ordres de vente).

```
double  SessionSellOrdersVolume() const
```

Valeur de retour

Le volume courant des ordres Sell (ordres de vente).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionOpen

Retourne le prix d'ouverture de la session courante.

```
double  SessionOpen() const
```

Valeur de retour

Le prix d'ouverture de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionClose

Retourne le prix de clôture de la session courante.

```
double  SessionClose() const
```

Valeur de retour

Le prix de clôture de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionAW

Retourne le prix moyen pondéré de la session courante.

```
double  SessionAW() const
```

Valeur de retour

Le prix moyen pondéré de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionPriceSettlement

Retourne le prix de compensation (settlement price) de la session courante.

```
double  SessionPriceSettlement() const
```

Valeur de retour

Le prix de compensation (settlement price) de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionPriceLimitMin

Retourne le prix minimal de la session courante.

```
double  SessionPriceLimitMin() const
```

Valeur de retour

Le prix minimal de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

SessionPriceLimitMax

Retourne le prix maximal de la session courante.

```
double  SessionPriceLimitMax() const
```

Valeur de retour

Le prix maximal de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
bool  InfoInteger(
   ENUM_SYMBOL_INFO_INTEGER  prop_id,      // Identifiant de la propriété
   long&                     var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer (valeur de l'énumération ENUM_SYMBOL_INFO_INTEGER<t3>).</t3>

var

[out] Référence à une variable de type long pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
bool  InfoDouble(
   ENUM_SYMBOL_INFO_DOUBLE  prop_id,      // Identifiant de la propriété
   double&                  var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double (valeur de l'énumération ENUM_SYMBOL_INFO_DOUBLE<t3>).</t3>

var

[out] Référence à une variable de type double pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
bool  InfoString(
   ENUM_SYMBOL_INFO_STRING  prop_id,      // Identifiant de la propriété
   string&                  var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété texte.

var

[out] Référence à une variable de type string pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

NormalizePrice

Retourne la valeur du prix, normalisée en utilisant les propriétés du symbole.

```
double  NormalizePrice(
   double  price   // prix
   ) const
```

Paramètres

price

[in] Prix.

Valeur de retour

Prix normalisé.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2024, MetaQuotes Ltd.

COrderInfo

COrderInfo est la classe d'accès aux propriétés des ordres en attente.

Description

La classe COrderInfo permet d'accéder aux propriétés d'un ordre en attente.

Déclaration

```
class COrderInfo : public CObject
```

Titre

```
#include <Trade\OrderInfo.mqh>
```

Hiérarchie d'héritage

CObject
  COrderInfo

Méthodes de classe par groupes

Accès aux propriétés de type integer	
Ticket	Retourne le ticket d'un ordre sélectionné auparavant
TimeSetup	Retourne l'heure de placement d'un ordre
TimeSetupMsc	Retourne l'heure de placement d'un ordre en millisecondes depuis le 01/01/1970
OrderType	Retourne le type de l'ordre
OrderTypeDescription	Retourne le type de l'ordre sous forme d'une chaîne de caractères
Etat	Retourne l'état de l'ordre
StateDescription	Retourne l'état de l'ordre sous forme d'une chaîne de caractères
TimeExpiration	Retourne l'heure d'expiration d'un ordre
TimeDone	Retourne l'heure d'exécution ou d'annulation d'un ordre
TimeDoneMsc	Retourne l'heure d'exécution ou d'annulation d'un ordre en millisecondes depuis le 01/01/1970
TypeFilling	Retourne le type d'exécution de l'ordre par ordre
TypeFillingDescription	Retourne le type d'exécution de l'ordre par ordre sous forme d'une chaîne de caractères
TypeTime	Retourne le type de l'ordre à l'heure d'expiration

© 2000-2024, MetaQuotes Ltd.

Accès aux propriétés de type integer	
TypeTimeDescription	Retourne le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères
Magic	Retourne l'identifiant de l'expert ayant placé l'ordre
PositionId	Retourne l'identifiant de la position
Accès aux propriétés de type double	
VolumeInitial	Retourne le volume initial de l'ordre
VolumeCurrent	Retourne le volume non atteint de l'ordre
PriceOpen	Retourne le prix de l'ordre
StopLoss	Retourne le Stop Loss de l'ordre
TakeProfit	Retourne le Take Profit de l'ordre
PriceCurrent	Retourne le prix courant du symbole
PriceStopLimit	Retourne le prix de l'ordre limite
Accès aux propriétés de type texte	
Symbol	Retourne le nom du symbole de l'ordre.
Comment	Retourne le commentaire de l'ordre
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string
Etat	
StoreState	Sauvegarde les paramètres de l'ordre
CheckState	Vérifie les paramères actuels par rapport aux paramètres sauvegardés
Sélection	
Select	Sélectionne un ordre par son ticket pour accéder par la suite à ses propriétés.
SelectByIndex	Sélectionne un ordre par son index pour accéder par la suite à ses propriétés.

Méthodes héritées de la classe CObject

© 2000-2024, MetaQuotes Ltd.

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Ticket

Retourne le ticket d'un ordre sélectionné précédemment en utilisant la méthode Select.

```
ulong  Ticket() const
```

Valeur de retour

Ticket de l'ordre en cas de succès, sinon ULONG_MAX.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeSetup

Retourne l'heure de placement de l'ordre.

```
datetime  TimeSetup() const
```

Valeur de retour

L'heure de placement de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeSetupMsc

Retourne l'heure de placement d'un ordre en millisecondes depuis le 01/01/1970<t2>.</t2>

```
ulong  TimeSetupMsc() const
```

Valeur de retour

L'heure de placement d'un ordre en millisecondes depuis le 01/01/1970.

Note

L'ordre doit tout d'abord être sélectionné pour pouvoir y accéder en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

OrderType

Retourne le type de l'ordre.

```
ENUM_ORDER_TYPE  OrderType()
```

Valeur de retour

Type de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE).

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeDescription

Retourne le type de l'ordre sous forme d'une chaîne de caractères.

```
string  TypeDescription() const
```

Valeur de retour

Type de l'ordre sous forme d'une chaîne de caractères.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Etat

Retourne l'état de l'ordre.

```
ENUM_ORDER_STATE  State() const
```

Valeur de retour

Etat de l'ordre (valeur de l'énumération ENUM_ORDER_STATE).

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

StateDescription

Retourne l'état de l'ordre sous forme d'une chaîne de caractères.

```
string  StateDescription() const
```

Valeur de retour

Etat de l'ordre sous forme d'une chaîne de caractères.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeExpiration

Retourne la date/heure d'expiration de l'ordre.

```
datetime  TimeExpiration() const
```

Valeur de retour

Date/heure d'expiration de l'ordre défini au moment du placement de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeDone

Retourne l'heure d'exécution ou d'annulation d'un ordre.

```
datetime  TimeDone() const
```

Valeur de retour

L'heure d'exécution ou d'annulation d'un ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeDoneMsc

Retourne l'heure d'exécution ou d'annulation d'un ordre en millisecondes depuis le 01/01/1970.

```
ulong  TimeDoneMsc() const
```

Valeur de retour

L'heure d'exécution ou d'annulation de l'ordre en millisecondes depuis le 01/01/1970.

Note

L'ordre doit tout d'abord être sélectionné pour pouvoir y accéder en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeFilling

Retourne le type de remplissage de l'ordre.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Valeur de retour

Type de remplissage de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_FILLING).

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeFillingDescription

Retourne le type de remplissage de l'ordre sous forme d'une chaîne de caractères.

```
string  TypeFillingDescription() const
```

Valeur de retour

Le type de remplissage de l'ordre sous forme d'une chaîne de caractères.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeTime

Retourne le type de l'ordre à l'heure d'expiration.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Valeur de retour

Le type de l'ordre àl'heure d'expiration.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeTimeDescription

Retourne le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères.

```
string  TypeTimeDescription() const
```

Valeur de retour

Le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Magic

Retourne l'identifiant de l'expert ayant placé l'ordre.

```
long  Magic() const
```

Valeur de retour

L'identifiant de l'expert ayant placé l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PositionId

Retourne l'identifiant de la position.

```
long  PositionId() const
```

Valeur de retour

Identifiant de la position dans laquelle l'ordre est impliqué.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

VolumeInitial

Retourne le volume initial de l'ordre.

```
double  VolumeInitial() const
```

Valeur de retour

Le volume initial de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

VolumeCurrent

Retourne le volume non atteint de l'ordre.

```
double  VolumeCurrent() const
```

Valeur de retour

Le volume non atteint de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PriceOpen

Retourne le prix de l'ordre.

```
double  PriceOpen() const
```

Valeur de retour

Prix de placement de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

StopLoss

Retourne le Stop Loss de l'ordre.

```
double  StopLoss() const
```

Valeur de retour

Le Stop Loss de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TakeProfit

Retourne le Take Profit de l'ordre.

```
double  TakeProfit() const
```

Valeur de retour

Le Take Profit de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PriceCurrent

Retourne le prix courant du symbole

```
double  PriceCurrent() const
```

Valeur de retour

Le prix courant du symbole

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PriceStopLimit

Retourne le prix de l'ordre limite.

```
double  PriceStopLimit() const
```

Valeur de retour

Le prix de l'ordre limite.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Symbol

Retourne le nom du symbole de l'ordre.

```
string  Symbol() const
```

Valeur de retour

Nom du symbole de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Comment

Retourne le commentaire de l'ordre

```
string  Comment() const
```

Valeur de retour

Commentaire de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // Identifiant de la propriété
   long&                        var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer (valeur de l'énumération ENUM_ORDER_PROPERTY_INTEGER).

var

[out] Référence à une variable de type long pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // Identifiant de la propriété
   double&                     var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double (valeur de l'énumération ENUM_ORDER_PROPERTY_DOUBLE).

var

[out] Référence à une variable de type double pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // Identifiant de la propriété
   string&                     var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété texte.

var

[out] Référence à une variable de type string pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

StoreState

Sauvegarde les paramètres de l'ordre.

```
void  StoreState()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CheckState

Vérifie les paramères actuels par rapport aux paramètres sauvegardés.

```
bool  CheckState()
```

Valeur de retour

vrai si les paramètres de l'ordre ont changé depuis le dernier appel à la méthode StoreState() , sinon faux.

© 2000-2024, MetaQuotes Ltd.

Select

Sélectionne un ordre par son ticket pour accéder par la suite à ses propriétés.

```
bool  Select(
   ulong  ticket   // ticket de l'ordre
   )
```

Valeur de retour

vrai en cas de succès, faux si l'ordre n'a pas pu être sélectionné.

© 2000-2024, MetaQuotes Ltd.

SelectByIndex

Sélectionne un ordre par son index pour accéder par la suite à ses propriétés.

```
bool  SelectByIndex(
   int  index   // index de l'ordre
   )
```

Valeur de retour

vrai en cas de succès, faux si l'ordre n'a pas pu être sélectionné.

© 2000-2024, MetaQuotes Ltd.

CHistoryOrderInfo

La classe CHistoryOrderInfo permet d'accéder aux propriétés d'un ordre de l'historique.

Description

La classe CHistoryOrderInfo permet d'accéder aux propriétés d'un ordre de l'historique.

Déclaration

```
class CHistoryOrderInfo : public CObject
```

Titre

```
#include <Trade\HistoryOrderInfo.mqh>
```

Hiérarchie d'héritage

CObject
  CHistoryOrderInfo

Méthodes de classe par groupes

Accès aux propriétés de type integer	
TimeSetup	Retourne l'heure de placement d'un ordre
TimeSetupMsc	Retourne l'heure de placement d'un ordre en millisecondes depuis le 01/01/1970
OrderType	Retourne le type de l'ordre
OrderTypeDescription	Retourne le type de l'ordre sous forme d'une chaîne de caractères
Etat	Retourne l'état de l'ordre
StateDescription	Retourne l'état de l'ordre sous forme d'une chaîne de caractères
TimeExpiration	Retourne l'heure d'expiration d'un ordre
TimeDone	Retourne l'heure d'exécution ou d'annulation d'un ordre
TimeDoneMsc	Retourne l'heure d'exécution ou d'annulation d'un ordre en millisecondes depuis le 01/01/1970
TypeFilling	Retourne le type d'exécution de l'ordre par ordre
TypeFillingDescription	Retourne le type d'exécution de l'ordre par ordre sous forme d'une chaîne de caractères
TypeTime	Retourne le type de l'ordre à l'heure d'expiration
TypeTimeDescription	Retourne le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères

© 2000-2024, MetaQuotes Ltd.

Accès aux propriétés de type integer	
Magic	Retourne l'identifiant de l'expert ayant placé l'ordre
PositionId	Retourne l'identifiant de la position
Accès aux propriétés de type double	
VolumeInitial	Retourne le volume initial de l'ordre
VolumeCurrent	Retourne le volume non atteint de l'ordre
PriceOpen	Retourne le prix de l'ordre
StopLoss	Retourne le Stop Loss de l'ordre
TakeProfit	Retourne le Take Profit de l'ordre
PriceCurrent	Retourne le prix courant du symbole
PriceStopLimit	Retourne le prix de l'ordre limite
Accès aux propriétés de type texte	
Symbol	Retourne le symbole de l'ordre
Comment	Retourne le commentaire de l'ordre
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string
Sélection	
Ticket	Retourne le ticket/sélectionne l'ordre
SelectByIndex	Sélectionne l'ordre par son index

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

TimeSetup

Retourne l'heure de placement de l'ordre.

```
datetime  TimeSetup() const
```

Valeur de retour

L'heure de placement de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeSetupMsc

Retourne l'heure de placement d'un ordre en millisecondes depuis le 01/01/1970.

```
ulong  TimeSetupMsc() const
```

Valeur de retour

L'heure de placement d'un ordre en millisecondes depuis le 01/01/1970

Note

L'ordre historique doit tout d'abord être sélectionné pour pouvoir y accéder en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

OrderType

Retourne le type de l'ordre.

```
ENUM_ORDER_TYPE  OrderType() const
```

Valeur de retour

Type de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE).

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeDescription

Retourne le type de l'ordre sous forme d'une chaîne de caractères.

```
string  TypeDescription() const
```

Valeur de retour

Type de l'ordre sous forme d'une chaîne de caractères.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Etat

Retourne l'état de l'ordre.

```
ENUM_ORDER_STATE  State() const
```

Valeur de retour

Etat de l'ordre (valeur de l'énumération ENUM_ORDER_STATE).

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

StateDescription

Retourne l'état de l'ordre sous forme d'une chaîne de caractères.

```
string  StateDescription() const
```

Valeur de retour

Etat de l'ordre sous forme d'une chaîne de caractères.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeExpiration

Retourne l'heure d'expiration d'un ordre.

```
datetime  TimeExpiration() const
```

Valeur de retour

L'heure d'expiration d'un ordre, définit lors de son placement.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeDone

Retourne l'heure d'exécution ou d'annulation d'un ordre.

```
datetime  TimeDone() const
```

Valeur de retour

L'heure d'exécution ou d'annulation d'un ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeDoneMsc

Retourne l'heure d'exécution ou d'annulation d'un ordre en millisecondes depuis le 01/01/1970.

```
ulong  TimeDoneMsc() const
```

Valeur de retour

L'heure d'exécution ou d'annulation de l'ordre en millisecondes depuis le 01/01/1970.

Note

L'ordre historique doit tout d'abord être sélectionné pour pouvoir y accéder en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeFilling

Retourne le type d'exécution de l'ordre par ordre.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Valeur de retour

Type de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_FILLING).

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeFillingDescription

Retourne le type d'exécution de l'ordre par ordre sous forme d'une chaîne de caractères.

```
string  TypeFillingDescription() const
```

Valeur de retour

Le type d'exécution de l'ordre par ordre sous forme d'une chaîne de caractères.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeTime

Retourne le type de l'ordre à l'heure d'expiration.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Valeur de retour

Type de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeTimeDescription

Retourne le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères.

```
string  TypeTimeDescription() const
```

Valeur de retour

Le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Magic

Retourne l'identifiant de l'Expert Advisor ayant placé l'ordre.

```
long  Magic() const
```

Valeur de retour

L'identifiant de l'Expert Advisor ayant placé l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PositionId

Retourne l'identifiant de la position.

```
long  PositionId() const
```

Valeur de retour

Identifiant de la position dans laquelle l'ordre est impliqué.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

VolumeInitial

Retourne le volume initial de l'ordre.

```
double  VolumeInitial() const
```

Valeur de retour

Le volume initial de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

VolumeCurrent

Retourne le volume non atteint de l'ordre.

```
double  VolumeCurrent() const
```

Valeur de retour

Le volume non atteint de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PriceOpen

Retourne le prix de l'ordre.

```
double  PriceOpen() const
```

Valeur de retour

Prix de placement de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

StopLoss

Retourne le prix du Stop Loss de l'ordre.

```
double  StopLoss() const
```

Valeur de retour

Le prix du Stop Loss de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TakeProfit

Retourne le prix du Take Profit de l'ordre.

```
double  TakeProfit() const
```

Valeur de retour

Le prix du Take Profit de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PriceCurrent

Retourne le prix courant du symbole de l'ordre.

```
double  PriceCurrent() const
```

Valeur de retour

Le prix courant du symbole de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PriceStopLimit

Retourne le prix stop limit de l'ordre.

```
double  PriceStopLimit() const
```

Valeur de retour

Prix stop limit de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Symbol

Retourne le nom du symbole de l'ordre.

```
string  Symbol() const
```

Valeur de retour

Nom du symbole de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Comment

Retourne le commentaire de l'ordre

```
string  Comment() const
```

Valeur de retour

Commentaire de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // Identifiant de la propriété
   long&                        var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer (valeur de l'énumération ENUM_ORDER_PROPERTY_INTEGER).

var

[out] Référence à une variable de type long pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // Identifiant de la propriété
   double&                     var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double (valeur de l'énumération ENUM_ORDER_PROPERTY_DOUBLE).

var

[out] Référence à une variable de type double pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // Identifiant de la propriété
   string&                     var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type texte (valeur de l'énumération ENUM_ORDER_PROPERTY_STRING).

var

[out] Référence à une variable de type string pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Ticket (Méthode "Get")

Retourne le ticket de l'ordre.

```
ulong  Ticket() const
```

Valeur de retour

Ticket de l'ordre.

Ticket (Méthode "Set")

Sélectionne l'ordre pour pouvoir l'utiliser ensuite.

```
void  Ticket(
   ulong  ticket      // ticket de l'ordre
   )
```

Paramètres

ticket

[in] Ticket de l'ordre.

© 2000-2024, MetaQuotes Ltd.

SelectByIndex

Sélectionne un ordre par son index pour accéder par la suite à ses propriétés.

```
bool  SelectByIndex(
   int  index   // index de l'ordre
   )
```

Valeur de retour

vrai en cas de succès, faux si l'ordre n'a pas pu être sélectionné.

© 2000-2024, MetaQuotes Ltd.

CPositionInfo

La classe CPositionInfo permet d'accéder aux propriétés des positions ouvertes.

Description

La classe CPositionInfo permet d'accéder aux propriétés des positions ouvertes.

Déclaration

```
class CPositionInfo : public CObject
```

Titre

```
#include <Trade\PositionInfo.mqh>
```

Hiérarchie d'héritage

CObject
  CPositionInfo

Méthodes de classe par groupes

Accès aux propriétés de type integer	
Time	Retourne l'heure de placement de l'ordre
TimeMsc	Retourne l'heure de placement de la position en millisecondes depuis le 01/01/1970
TimeUpdate	Retourne l'heure de changement de la position en secondes depuis le 01/01/1970
TimeUpdateMsc	Retourne l'heure de changement de la position en millisecondes depuis le 01/01/1970
PositionType	Retourne le type de la position
TypeDescription	Retourne le type de la position sous forme d'une chaîne de caractères
Magic	Retourne l'identifiant de l'expert ayant ouvert la position
Identifiant	Retourne l'identifiant de la position
Accès aux propriétés de type double	
Volume	Retourne le volume de la position
PriceOpen	Retourne le prix d'ouverture de la position
StopLoss	Retourne le prix du Stop Loss de la position
TakeProfit	Retourne le prix du Take Profit de la position

© 2000-2024, MetaQuotes Ltd.

Accès aux propriétés de type integer	
PriceCurrent	Retourne le prix actuel du symbole de la position
Commission	Retourne le montant de la commission de la position
Swap	Retourne le montant du swap de la position
Profit	Retourne le montant du profit actuel de la position
Accès aux propriétés de type texte	
Symbol	Retourne le nom du symbole de la position
Comment	Retourne le commentaire de la position
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string.
Sélection	
Select	Sélectionne la position.
SelectByIndex	Sélectionne la position par son index
SelectByMagic	Sélectionne une position avec le symbole et le nombre magique spécifiés
SelectByTicket	Sélectionne une position par son ticket
Etat	
StoreState	Sauvegarde les paramètres de la position
CheckState	Vérifie les paramères actuels par rapport aux paramètres sauvegardés

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Time

Retourne l'heure d'ouverture de la position.

```
datetime  Time() const
```

Valeur de retour

L'heure d'ouverture de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeMsc

Retourne l'heure d'ouverture de la position en millisecondes depuis le 01/01/1970.

```
ulong  TimeMsc() const
```

Valeur de retour

L'heure d'ouverture de la position en millisecondes depuis le 01/01/1970.

Note

La position doit d'abord être sélectionnée en utilisant les méthodes Select (par le symbole) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeUpdate

Retourne l'heure de changement de la position en secondes depuis le 01/01/1970.

```
datetime  TimeUpdate() const
```

Valeur de retour

L'heure de changement de la position en secondes depuis le 01/01/1970.

Note

La position doit d'abord être sélectionnée en utilisant les méthodes Select (par le symbole) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TimeUpdateMsc

Retourne l'heure de changement de la position en millisecondes depuis le 01/01/1970.

```
ulong  TimeUpdateMsc() const
```

Valeur de retour

L'heure de changement de la position en millisecondes depuis le 01/01/1970.

Note

La position doit d'abord être sélectionnée en utilisant les méthodes Select (par le symbole) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PositionType

Retourne le type de la position.

```
ENUM_POSITION_TYPE  PositionType() const
```

Valeur de retour

Type de la position (valeur de l'énumération ENUM_POSITION_TYPE).

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TypeDescription

Retourne le type de la position sous forme d'une chaîne de caractères.

```
string  TypeDescription() const
```

Valeur de retour

Le type de la position sous forme d'une chaîne de caractères.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Magic

Retourne l'identifiant de l'Expert Advisor ayant ouvert la position.

```
long  Magic() const
```

Valeur de retour

L'identifiant de l'Expert Advisor ayant ouvert la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Identifier

Retourne l'identifiant de la position.

```
long  Identifier() const
```

Valeur de retour

L'identifiant de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Volume

Retourne le volume de la position.

```
double  Volume() const
```

Valeur de retour

Le volume de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PriceOpen

Retourne le prix d'ouverture de la position.

```
double  PriceOpen() const
```

Valeur de retour

Le prix d'ouverture de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

StopLoss

Retourne le prix du Stop Loss de la position.

```
double  StopLoss() const
```

Valeur de retour

Le prix du Stop Loss de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

TakeProfit

Retourne le prix du Take Profit de la position.

```
double  TakeProfit() const
```

Valeur de retour

Le prix du Take Profit de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

PriceCurrent

Retourne le prix actuel du symbole de la position.

```
double  PriceCurrent() const
```

Valeur de retour

Le prix actuel du symbole de la position.

© 2000-2024, MetaQuotes Ltd.

Commission

Retourne le montant de la commission de la position.

```
double  Commission() const
```

Valeur de retour

Le montant de la commission de la position (dans la devise du compte).

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Swap

Retourne le montant du swap de la position.

```
double  Swap() const
```

Valeur de retour

Le montant du swap de la position (dans la devise du compte).

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Profit

Retourne le montant du profit actuel de la position.

```
double  Profit() const
```

Valeur de retour

Le montant du profit actuel de la position (dans la devise du compte).

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Symbol

Retourne le nom du symbole de la position.

```
string  Symbol() const
```

Valeur de retour

Le nom du symbole de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Comment

Retourne le commentaire de la position.

```
string  Comment() const
```

Valeur de retour

Le commentaire de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
bool  InfoInteger(
   ENUM_POSITION_PROPERTY_INTEGER  prop_id,      // Identifiant de la propriété
   long&                           var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer (valeur de l'énumération ENUM_POSITION_PROPERTY_INTEGER).

var

[out] Référence à une variable de type long pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
bool  InfoDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  prop_id,      // Identifiant de la propriété
   double&                        var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double (valeur de l'énumération ENUM_POSITION_PROPERTY_DOUBLE).

var

[out] Référence à une variable de type double pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
bool  InfoString(
   ENUM_POSITION_PROPERTY_STRING  prop_id,      // Identifiant de la propriété
   string&                        var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type texte (valeur de l'énumération ENUM_POSITION_PROPERTY_STRING).

var

[out] Référence à une variable de type string pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Select

Sélectionne la position pour pouvoir l'utiliser ensuite.

```
bool  Select(
   const string  symbol      // symbole
   )
```

Paramètres

symbol

[in] Symbole pour la sélection de la position.

© 2000-2024, MetaQuotes Ltd.

SelectByIndex

Sélectionne la position par son index pour accéder par la suite à ses propriétés.

```
bool  SelectByIndex(
   int  index   // index de la position
   );
```

Valeur de retour

vrai en cas de succès, faux si la position n'a pas pu être sélectionnée.

© 2000-2024, MetaQuotes Ltd.

SelectByMagic

Sélectionne une position sur la base du nom de l'instrument financier et le nombre magique pour pouvoir travailler avec elle.

```
bool  SelectByMagic(
   const string  symbol, // Nom du symbole
   const ulong   magic   // Nombre magique
   );
```

Paramètres

symbol

[in] Nom du symbole.

magic

[in] Nombre magique de la position.

Valeur de Retour

Retourne true en cas de succès ou false si la sélection de la position a échoué.

© 2000-2024, MetaQuotes Ltd.

SelectByTicket

Sélectionne une position par son ticket pour pouvoir travailler avec elle.

```
bool  SelectByTicket(
   ulong  ticket      // ticket de la position
   );
```

Paramètres

ticket

[in] Ticket de la position.

Valeur de Retour

true - en cas de succès, false - aucune position sélectionnée.

© 2000-2024, MetaQuotes Ltd.

StoreState

Sauvegarde les paramètres de la position.

```
void  StoreState()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CheckState

Vérifie les paramères actuels par rapport aux paramètres sauvegardés.

```
bool  CheckState()
```

Valeur de retour

vrai si les paramètres de la position ont changé depuis le dernier appel à la méthode StoreState() , sinon faux.

© 2000-2024, MetaQuotes Ltd.

CDealInfo

La classe CDealInfo permet d'accéder aux propriétés du deal.

Description

La classe CDealInfo fournit l'accès aux propriétés du deal.

Déclaration

```
class CDealInfo : public CObject
```

Titre

```
#include <Trade\DealInfo.mqh>
```

Hiérarchie d'héritage

CObject
  CDealInfo

Méthodes de classe par groupes

Accès aux propriétés de type integer	
Order	Retourne l'ordre correspondant au deal exécuté
Time	Retourne l'heure d'exécution du deal
TimeMsc	Retourne l'heure d'exécution d'un deal en millisecondes depuis le 01/01/1970
DealType	Retourne le type du deal
TypeDescription	Retourne le type du deal sous forme d'une chaîne de caractères
Entry	Retourne la direction du deal
EntryDescription	Retourne la direction du deal sous forme d'une chaîne de caractères
Magic	Retourne l'identifiant de l'expert ayant exécuté le deal
PositionId	Retourne l'identifiant de la position dans laquelle le deal est impliqué
Accès aux propriétés de type double	
Volume	Retourne le volume du deal
Price	Retourne le prix du deal
Commision	Retourne le montant de commission par deal

© 2000-2024, MetaQuotes Ltd.

Accès aux propriétés de type integer	
Swap	Retourne le montant du swap lorsque la position est clôturée
Profit	Retourne le résultat financier du deal
Accès aux propriétés de type texte	
Symbol	Retourne le nom du symbole du deal
Comment	Retourne le commentaire du deal
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string
Sélection	
Ticket	Retourne le ticket/sélectionne le deal
SelectByIndex	Sélectionne le deal par son index

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Order

Retourne l'ordre correspondant au deal exécuté.

```
long  Order() const
```

Valeur de retour

Ordre correspondant au deal exécuté.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket **(par le ticket) ou** SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

Time

Retourne l'heure d'exécution du deal.

```
datetime  Time() const
```

Valeur de retour

L'heure d'exécution du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

TimeMsc

Retourne l'heure d'exécution d'un deal en millisecondes depuis le 01/01/1970.

```
ulong  TimeMsc() const
```

Valeur de retour

L'heure d'exécution d'un deal en millisecondes depuis le 01/01/1970.

Note

Le deal doit tout d'abord être sélectionné pour pouvoir y accéder en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2024, MetaQuotes Ltd.

DealType

Retourne le type du deal.

```
ENUM_DEAL_TYPE  DealType() const
```

Valeur de retour

Type du deal (valeur de l'énumération ENUM_DEAL_TYPE).

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

TypeDescription

Retourne le type du deal sous forme d'une chaîne de caractères.

```
string  TypeDescription() const
```

Valeur de retour

Type du deal sous forme d'une chaîne de caractères.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

Entry

Retourne la direction du deal.

```
ENUM_DEAL_ENTRY  Entry() const
```

Valeur de retour

Direction du deal (valeur de l'énumération ENUM_DEAL_ENTRY).

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

EntryDescription

Retourne la direction du deal sous forme d'une chaîne de caractères.

```
string  EntryDescription() const
```

Valeur de retour

Direction du deal sous forme d'une chaîne de caractères.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

Magic

Retourne l'identifiant de l'Expert Advisor ayant exécuté le deal.

```
long  Magic() const
```

Valeur de retour

Identifiant de l'Expert Advisor ayant exécuté le deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

PositionId

Retourne l'identifiant de la position dans laquelle le deal est impliqué.

```
long  PositionId() const
```

Valeur de retour

Identifiant de la position dans laquelle le deal est impliqué.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

Volume

Retourne le volume du deal.

```
double  Volume() const
```

Valeur de retour

Volume du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

Price

Retourne le prix du deal.

```
double  Price() const
```

Valeur de retour

Prix du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

Commission

Retourne le montant de commission du deal.

```
double  Commission() const
```

Valeur de retour

Le montant de commission du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

Swap

Retourne le montant du swap lorsque la position est clôturée.

```
double  Swap() const
```

Valeur de retour

Montant du swap lorsque la position est clôturée.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

Profit

Retourne le résultat financier du deal.

```
double  Profit() const
```

Valeur de retour

Résultat financier du deal (dans la devise du compte).

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

Symbol

Retourne le nom du symbole du deal.

```
string  Symbol() const
```

Valeur de retour

Le nom du symbole du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

Comment

Retourne le commentaire du deal.

```
string  Comment() const
```

Valeur de retour

Commentaire du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
bool  InfoInteger(
   ENUM_DEAL_PROPERTY_INTEGER  prop_id,      // Identifiant de la propriété
   long&                       var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer (valeur de l'énumération ENUM_DEAL_PROPERTY_INTEGER).

var

[out] Référence à une variable de type long pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
bool  InfoDouble(
   ENUM_DEAL_PROPERTY_DOUBLE  prop_id,      // Identifiant de la propriété
   double&                    var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double (valeur de l'énumération ENUM_DEAL_PROPERTY_DOUBLE).

var

[out] Référence à une variable de type double pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
bool  InfoString(
   ENUM_DEAL_PROPERTY_STRING  prop_id,      // Identifiant de la propriété
   string&                    var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type texte (valeur de l'énumération ENUM_DEAL_PROPERTY_STRING).

var

[out] Référence à une variable de type string pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2024, MetaQuotes Ltd.

Ticket (Méthode "Get")

Retourne le ticket du deal.

```
ulong  Ticket() const
```

Valeur de retour

Ticket du deal.

Ticket (Méthode "Set")

Sélectionne la position pour pouvoir l'utiliser ensuite.

```
void  Ticket(
   ulong  ticket      // ticket
   )
```

Paramètres

ticket

[in] Ticket du deal.

© 2000-2024, MetaQuotes Ltd.

SelectByIndex

Sélectionne le deal par son index pour accéder par la suite à ses propriétés.

```
bool  SelectByIndex(
   int  index   // index de l'ordre
   )
```

Valeur de retour

vrai en cas de succès, faux si le deal n'a pas pu être sélectionné.

© 2000-2024, MetaQuotes Ltd.

CTrade

La classe CTrade est une classe permettant d'accéder aux fonctions de trading.

Description

La classe CTrade fournit l'accès aux fonctions de trading.

Déclaration

```
class CTrade : public CObject
```

Titre

```
#include <Trade\Trade.mqh>
```

Hiérarchie d'héritage

CObject
  CTrade

Descendants directs

CExpertTrade

Méthodes de classe par groupes

Définition des paramètres	
LogLevel	Définit le niveau de journalisation
SetExpertMagicNumber	Définit l'identifiant de l'expert
SetDeviationInPoints	Définit la déviation autorisée
SetTypeFilling	Définit le type de remplissage de l'ordre
SetTypeFillingBySymbol	Définit le fillingtype de l'ordre selon les paramètres du symbole spécifié
SetAsyncMode	Définit le mode asynchrone pour les opérations de trading
SetMarginMode	Sets margin calculation mode in accordance with the current account settings
Opérations sur les ordres	
OrderOpen	Place un ordre en attente avec les paramètres définis
OrderModify	Modifie les paramètres d'un ordre en attente
OrderDelete	Supprime un ordre en attente
Opérations sur les positions	
PositionOpen	Ouvre une position avec les paramètres définis

© 2000-2024, MetaQuotes Ltd.

Définition des paramètres	
PositionModify	Modifie les paramètres d'une position
PositionClose	Clôture une position
PositionClosePartial	Ferme partiellement une position sur un symbole spécifié ou ayant un ticket spécifié
PositionCloseBy	Ferme une position avec le ticket spécifié par une position opposée
Méthodes supplémentaires	
Buy	Ouvre une position longue avec les paramètres spécifiés
Sell	Ouvre une position courte avec les paramètres spécifiés
BuyLimit	Place un ordre en attente de type Buy Limit avec les paramètres spécifiés
BuyStop	Place un ordre en attente de type Buy Stop avec les paramètres spécifiés
SellLimit	Place un ordre en attente de type Sell Limit avec les paramètres spécifiés
SellStop	Place un ordre en attente de type Sell Stop avec les paramètres spécifiés
Accès aux paramètres de la dernière requête	
Request	Retourne une copie de la structure de la dernière requête
RequestAction	Retourne le type d'une opération de trading
RequestActionDescription	Retourne le type d'une opération de trading sous forme d'une chaîne de caractères
RequestMagic	Retourne l'identifiant Magic de l'Expert Advisor
RequestOrder	Retourne le ticket de l'ordre utilisé dans la dernière requête
RequestSymbol	Retourne le nom du symbole utilisé dans la dernière requête
RequestVolume	Retourne le volume (en lots) utilisé dans la dernière requête
RequestPrice	Retourne le prix utilisé dans la dernière requête
RequestStopLimit	Retourne le prix d'un ordre en attente de type Stop Limit, utilisé dans la dernière requête
RequestSL	Retourne le prix du Stop Loss de l'ordre, utilisé dans la dernière requête
RequestTP	Retourne le prix du Take Profit de l'ordre, utilisé dans la dernière requête

© 2000-2024, MetaQuotes Ltd.

Définition des paramètres	
RequestDeviation	Retourne la déviation du prix de l'ordre, utilisé dans la dernière requête
RequestType	Retourne le type de l'ordre, utilisé dans la dernière requête
RequestTypeDescription	Retourne le type de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête
RequestTypeFilling	Retourne le type de remplissage de l'ordre, utilisé dans la dernière requête
RequestTypeFillingDescription	Retourne le type de remplissage de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête
RequestTypeTime	Retourne la période de validité de l'ordre, utilisé dans la dernière requête
RequestTypeTimeDescription	Retourne la période de validité de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête
RequestExpiration	Retourne l'heure d'expiration de l'ordre utilisé dans la dernière requête
RequestComment	Retourne le commentaire de l'ordre utilisé dans la dernière requête
RequestPosition	Retourne le ticket de la position
RequestPositionBy	Retourne le ticket de la position opposée
Accès aux résultats de la dernière requête	
CheckResult	Retourne une copie de la structure des résultats de la dernière requête.
CheckResultRetcode	Retourne la valeur du champ retcode du type MqlTradeCheckResult, rempli lors de la vérification de l'exactitude de la requête
CheckResultRetcodeDescription	Retourne la description textuelle du champ retcode du type MqlTradeCheckResult, rempli lors de la vérification de l'exactitude de la requête
CheckResultBalance	Retourne la valeur du champ balance du type MqlTradeCheckResult, rempli lors de la vérification de l'exactitude de la requête
CheckResultEquity	Retourne la valeur du champ equity du type MqlTradeCheckResult, rempli lors de la vérification de l'exactitude de la requête
CheckResultProfit	Retourne la valeur du champ profit du type MqlTradeCheckResult, rempli lors de la vérification de l'exactitude de la requête
CheckResultMargin	Retourne la valeur du champ margin du type MqlTradeCheckResult, rempli lors de la vérification de l'exactitude

© 2000-2024, MetaQuotes Ltd.

Définition des paramètres	
	de la requête
CheckResultMarginFree	Retourne la valeur du champ margin_free du type MqlTradeCheckResult, rempli lors de la vérification de l'exactitude de la requête
CheckResultMarginLevel	Retourne la valeur du champ margin_level du type MqlTradeCheckResult, rempli lors de la vérification de l'exactitude de la requête
CheckResultComment	Retourne la valeur du champ comment du type MqlTradeCheckResult, rempli lors de la vérification de l'exactitude de la requête
Accès aux résultat de l'exécution de la dernière requête	
Result	Retourne une copie de la structure des résultats de la dernière requête.
ResultRetcode	Retourne le code du résultat de la requête
ResultRetcodeDescription	Retourne le code du résultat de la requête sous forme de texte
ResultDeal	Retourne le ticket du deal
ResultOrder	Retourne le ticket de l'ordre
ResultVolume	Retourne le volume du deal ou de l'ordre
ResultPrice	Retourne le prix confirmé par le courtier
ResultBid	Retourne le prix bid courant (la recotation)
ResultAsk	Retourne le prix ask courant (la recotation)
ResultComment	Retourne le commentaire du courtier
Autres méthodes	
PrintRequest	Ecrit les paramètres de la dernière requête dans le journal
PrintResult	Ecrit les résultats de la dernière requête dans le journal
FormatRequest	Prépare la chaîne de caractères formatée avec les paramètres de la dernière requête
FormatRequestResult	Prépare la chaîne de caractères formatée avec les résultats d'exécution de la dernière requête

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

LogLevel

Définit le niveau de journalisation des messages.

```
void  LogLevel(
   ENUM_LOG_LEVELS  log_level      // niveau de journalisation
   )
```

Paramètres

log_level

[in] Niveau de journalisation.

Valeur de retour

Aucune.

Note

LOG_LEVEL_NO et en-dessous désactive l'affichage de tous les messages (définit automatiquement dans le mode d'optimisation). LOG_LEVEL_ERRORS active l'affichage des messages d'erreurs uniquement (valeur par défaut). LOG_LEVEL_ALL et au-dessus active l'affichage de tous les messages (définit automatiquement dans le mode de test).

ENUM_LOG_LEVELS

Identifier	Description	Value
LOG_LEVEL_NO	L'affichage des messages est désactivé	0
LOG_LEVEL_ERRORS	Seuls les messages d'erreurs sont affichés	1
LOG_LEVEL_ALL	Tous les messages sont affichés	2

© 2000-2024, MetaQuotes Ltd.

SetExpertMagicNumber

Définit l'identifiant de l'expert.

```
void  SetExpertMagicNumber(
   ulong  magic      // Identifiant
   )
```

Paramètres

magic

[in] Nouvel identifiant de l'expert.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

SetDeviationInPoints

Définit la déviation autorisée.

```
void  SetDeviationInPoints(
   ulong  deviation      // Déviation
   )
```

Paramètres

deviation

[in] Déviation autorisée.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

SetTypeFilling

Définit le type de remplissage de l'ordre.

```
void  SetTypeFilling(
   ENUM_ORDER_TYPE_FILLING  filling      // type de remplissage de l'ordre
   )
```

Paramètres

filling

[in] Type de remplissage de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_FILLING).

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

SetTypeFillingBySymbol

Définit le fillingtype de l'ordre selon les paramètres du symbole spécifié.

```
bool  SetTypeFillingBySymbol(
   const string   symbol      // nom du symbole
   )
```

Paramètres

symbol

[in] Nom du symbole, pour lequel SYMBOL_FILLING_MODE contient les politiques de remplissage d'ordre autorisées.

Valeur de Retour

true - en cas d'exécution réussie, false - en cas d'échec de la définition de la politique de remplissage.

Note

Si les politiques de remplissage SYMBOL_FILLING_FOK et SYMBOL_FILLING_IOC sont autorisées pour un symbole en même temps, la valeur ORDER_FILLING_FOK est utilisée pour l'ordre.

© 2000-2024, MetaQuotes Ltd.

SetAsyncMode

Définit le mode asynchrone pour les opérations de trading.

```
void  SetAsyncMode(
   bool  mode      // flag du mode asynchrone
  )
```

Paramètres

mode

[in] Flag du mode asynchrone.

Valeur de retour

Aucune.

Note

Ce mode est utilisé pour les opérations de trading en mode asynchrone (sans attendre la réponse du serveur de trading à la requête envoyée) - voir OrderSendAsync.

© 2000-2024, MetaQuotes Ltd.

SetMarginMode

Définit le mode de calcul de la marge selon les paramètres actuels du compte.

```
void  SetMarginMode()
```

Valeur de Retour

Aucune.

Note

Le mode de calcul de la marge est spécifié dans ENUM_ACCOUNT_MARGIN_MODE.

© 2000-2024, MetaQuotes Ltd.

OrderOpen

Place un ordre en attente avec les paramètres définis.

```
bool  OrderOpen(
   const string          symbol,          // symbole
   ENUM_ORDER_TYPE       order_type,      // type d'ordre
   double                volume,          // volume de l'ordre
   double                limit_price,     // prix StopLimit
   double                price,           // prix d'exécution
   double                sl,              // prix du stop loss
   double                tp,              // prix du take profit
   ENUM_ORDER_TYPE_TIME  type_time,       // type par expiration
   datetime              expiration,      // expiration
   const string          comment=""       // commentaire
   )
```

Paramètres

symbol

[in] Nom de l'instrument de trading.

order_type

[in] Type de l'opération de trading (valeur de l'énumération ENUM_ORDER_TYPE).

volume

[in] Volume demandé pour l'ordre.

limit_price

[in] Prix auquel l'ordre StopLimit sera placé.

price

[in] Prix auquel l'ordre doit être exécuté.

sl

[in] Prix auquel le Stop Loss sera déclenché.

tp

[in] Prix auquel le Take Profit sera déclenché.

type_time

[in] Type de validité de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

expiration

[in] Date d'expiration de l'ordre en attente.

comment=""

[in] Commentaire de l'ordre.

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Note

La réussite de la méthode OrderSend(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par <li5>ResultOrder()</li5><t6>.</t6>******

© 2000-2024, MetaQuotes Ltd.

OrderModify

Modifie les paramètres d'un ordre en attente.

```
bool  OrderModify(
   ulong                ticket,         // ticket de l'ordre
   double               price,          // prix d'exécution
   double               sl,             // prix du stop loss
   double               tp,             // prix du take profit
   ENUM_ORDER_TYPE_TIME type_time,      // type par expiration
   datetime             expiration,     // expiration
   double               stoplimit       // Prix limite de l'ordre
   )
```

Paramètres

ticket

[in] Ticket de l'ordre.

price

[in] Le nouveau prix auquel l'ordre doit être exécuté (ou la valeur précédente si le changement n'est pas nécessaire).

sl

[in] Le nouveau prix auquel le Stop Loss sera déclenché (ou la valeur précédente si le changement n'est pas nécessaire).

tp

[in] Le nouveau prix auquel le Take Profit sera déclenché (ou la valeur précédente si le changement n'est pas nécessaire).

type_time

[in] Le nouveau type d'expiration de l'ordre (ou la valeur précédente si le changement n'est pas nécessaire), valueur de l'énumération ENUM_ORDER_TYPE_TIME.

expiration

[in] La nouvelle date d'expiration de l'ordre en attente (ou la valeur précédente si le changement n'est pas nécessaire).

stoplimit

[in] Le nouveau prix utilisé pour placer un ordre Limit lorsque le prix atteint la valeur du paramètre *price* . Il n'est spécifié que pour les ordres StopLimit.

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

Note

La réussite de la méthode OrderModify(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode().

© 2000-2024, MetaQuotes Ltd.

OrderDelete

Supprime un ordre en attente.

```
bool  OrderDelete(
   ulong  ticket      // ticket de l'ordre
   )
```

Paramètres

ticket

[in] Ticket de l'ordre.

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

Note

La réussite de la méthode OrderDelete(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode().

© 2000-2024, MetaQuotes Ltd.

PositionOpen

Ouvre une position avec les paramètres définis.

```
bool  PositionOpen(
   const string     symbol,          // symbole
   ENUM_ORDER_TYPE  order_type,      // type d'ordre to open position
   double           volume,          // volume de la position
   double           price,           // prix d'exécution
   double           sl,              // prix du stop loss
   double           tp,              // prix du take profit
   const string     comment=""       // commentaire
   )
```

Paramètres

symbol

[in] Nom de l'instrument de trading sur lequel la position doit être ouverte.

order_type

[in] Type d'ordre (opération de trading) pour ouvrir une position (valeur de l'énumération ENUM_ORDER_TYPE).

volume

[in] Volume demandé pour la position.

price

[in] Prix auquel la position doit être ouverte.

sl

[in] Prix auquel le Stop Loss sera déclenché.

tp

[in] Prix auquel le Take Profit sera déclenché.

comment=""

[in] Commentaire de la position.

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

Note

La réussite de la méthode PositionOpen(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par <li5>ResultDeal()</li5><t6>.</t6>

© 2000-2024, MetaQuotes Ltd.

PositionModify

Modifie les paramètres d'une position sur le symbole spécifié.

```
bool  PositionModify(
   const string  symbol,      // symbole
   double        sl,          // prix du stop loss
   double        tp           // prix du take profit
   )
```

Modifies les paramètres d'une position par le ticket spécifié.

```
bool  PositionModify(
   const ulong   ticket,      // ticket de la position
   double        sl,          // Prix du Stop Loss price
   double        tp           // Prix du Take Profit
   )
```

Paramètres

symbol

[in] Nom de l'instrument de trading sur lequel la position doit être modifiée.

ticket

[in] Ticket de la position à modifier.

sl

[in] Le nouveau prix auquel le Stop Loss sera déclenché (ou la valeur précédente si le changement n'est pas nécessaire).

tp

[in] Le nouveau prix auquel le Take Profit sera déclenché (ou la valeur précédente si le changement n'est pas nécessaire).

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

Note

La réussite de la méthode PositionModify(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode().

© 2000-2024, MetaQuotes Ltd.

PositionClose

Clôture une position sur le symbole spécifié.

```
bool  PositionClose(
   const string  symbol,                  // symbole
   ulong         deviation=ULONG_MAX      // Déviation
   )
```

Ferme une position avec le ticket spécifié.

```
bool  PositionClose(
   const ulong   ticket,                  // ticket de la position
   ulong         deviation=ULONG_MAX      // Déviation
   )
```

Paramètres

symbol

[in] Nom de l'instrument de trading sur lequel la position doit être clôturée.

ticket

[in] Ticket d'une position fermée.

deviation=ULONG_MAX

[in] Déviation maximale par rapport au prix courant (en points).

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

Note

La réussite de la méthode PositionClose(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode().

© 2000-2024, MetaQuotes Ltd.

PositionClosePartial

Ferme partiellement une position sur le symbole spécifié dans le cas d'un compte avec "hedging".

```
bool  PositionClosePartial(
   const string  symbol,                  // symbole
   const double  volume,                  // volume
   ulong         deviation=ULONG_MAX      // déviation
   )
```

Ferme partiellement une position ayant le ticket spécifié dans le cas d'un compte avec "hedging".

```
bool  PositionClosePartial(
   const ulong   ticket,                  // ticket de la position
   const double  volume,                  // volume
   ulong         deviation=ULONG_MAX      // déviation
   )
```

Paramètres

symbol

[in] Nom de l'instrument de trading sur lequel une position est fermée partiellement. Si un symbole (et non pas un ticket) est spécifié pour la fermeture partielle d'une position, la première position détectée ayant le MagicNumber spécifié (Identifiant de l'Expert Advisor) sur le symbole est sélectionnée. Il est donc parfois mieux d'utiliser PositionClosePartial() en spécifiant le ticket de la position.

volume

[in] Volume de la position devant être fermé. Si la valeur est supérieure au volume d'une position partiellement fermée, elle est fermée intégralement. Aucune position dans la direction opposée n'est ouverte.

ticket

[in] Ticket de la position fermée.

deviation=ULONG_MAX

[in] La déviation maximum du prix actuel (en points).

Valeur de Retour

true si les vérifications de base des structures sont passées, false sinon.

Note

Une exécution réussie de la méthode PositionClosePartial(...) ne signifie pas toujours une exécution réussie d'une opération de trading. Vous devriez appeler la méthode ResultRetcode() pour vérifier le résultat d'une demande de trade (code de retour du serveur de trades).

Dans le système "netting" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING et ACCOUNT_MARGIN_MODE_EXCHANGE), pour chaque symbole, à n'importe quel moment, seule une position peut être ouverte, qui est le résultat d'une ou plusieurs transactions. Ne confondez pas les ordres en attente actuels avec les positions qui sont également affichées dans l'onglet "Trade" du panneau "Boîte à Outils" du terminal client.

© 2000-2024, MetaQuotes Ltd.

Dans le cas de représentation de la position (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), plusieurs positions peuvent être ouvertes simultanément sur chaque symbole. Dans ce cas, PositionClose ferme la position ayant le dernier ticket.

© 2000-2024, MetaQuotes Ltd.

PositionCloseBy

Ferme une position avec le ticket spécifié par une position opposée.

```
bool  PositionCloseBy(
   const ulong   ticket,        // ticket de la position
   const ulong   ticket_by      // Ticket de la position opposée
   )
```

Paramètres

ticket

[in] Ticket de la position fermée.

ticket_by

[in] Ticket de la position opposée utilisée pour la fermeture.

Valeur de Retour

true si la vérification de base des structures est correcte, false sinon.

Note

Une fin avec succès de la méthode PositionCloseBy(...) ne signifie pas toujours une exécution réussie de l'opération de trading. Vous devez appeler la méthode ResultRetcode() pour vérifier le résultat d'une demande de trade (le code de retour du serveur de trading).

© 2000-2024, MetaQuotes Ltd.

Buy

Ouvre une position longue avec les paramètres spécifiés.

```
bool  Buy(
   double        volume,          // volume de la position
   const string  symbol=NULL,     // symbole
   double        price=0.0,       // prix
   double        sl=0.0,          // prix du stop loss
   double        tp=0.0,          // prix du take profit
   const string  comment=""       // commentaire
   )
```

Paramètres

volume

[in] Volume de la position.

symbol=NULL

[in] Symbole de la position. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

price=0.0

[in] Prix. Si le prix n'est pas spécifié, le prix Ask du marché courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

comment=""

[in] Commentaire.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode Buy(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultDeal().

© 2000-2024, MetaQuotes Ltd.

Sell

Ouvre une position courte avec les paramètres spécifiés.

```
bool  Sell(
   double        volume,          // volume de la position
   const string  symbol=NULL,     // symbole
   double        price=0.0,       // prix
   double        sl=0.0,          // prix du stop loss
   double        tp=0.0,          // prix du take profit
   const string  comment=""       // commentaire
   )
```

Paramètres

volume

[in] Volume de la position.

symbol=NULL

[in] Symbole de la position. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

price=0.0

[in] Prix. Si le prix n'est pas spécifié, le prix Bid du marché courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

comment=""

[in] Commentaire.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode Sell(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultDeal().

© 2000-2024, MetaQuotes Ltd.

BuyLimit

Place un ordre en attente de type Buy Limit (achat au prix, plus bas que le prix actuel du marché) avec les paramètres spécifiés.

```
bool  BuyLimit(
   double                volume,                      // volume de l'ordre
   double                price,                       // prix de l'ordre
   const string          symbol=NULL,                 // symbole
   double                sl=0.0,                      // prix du stop loss
   double                tp=0.0,                      // prix du take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,    // durée de vie de l'ordre
   datetime              expiration=0,                // heure d'expiration de l'ordr
   const string          comment=""                   // commentaire
   )
```

Paramètres

volume

[in] Volume de l'ordre.

price

[in] Prix de l'ordre.

symbol=NULL

[in] Symbole de l'ordre. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

type_time=ORDER_TIME_GTC

[in] Durée de vie de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Heure d'expiration de l'ordre (utilisé uniquement si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commentaire de l'ordre.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode BuyLimit(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultOrder().

© 2000-2024, MetaQuotes Ltd.

BuyStop

Place un ordre en attente de type Buy Stop (achat au prix, plus haut que le prix actuel du marché) avec les paramètres spécifiés.

```
bool  BuyStop(
   double                volume,                      // volume de l'ordre
   double                price,                       // prix de l'ordre
   const string          symbol=NULL,                 // symbole
   double                sl=0.0,                      // prix du stop loss
   double                tp=0.0,                      // prix du take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,    // durée de vie de l'ordre
   datetime              expiration=0,                // heure d'expiration de l'ordı
   const string          comment=""                   // commentaire
   )
```

Paramètres

volume

[in] Volume de l'ordre.

price

[in] Prix de l'ordre.

symbol=NULL

[in] Symbole de l'ordre. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

type_time=ORDER_TIME_GTC

[in] Durée de vie de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Heure d'expiration de l'ordre (utilisé uniquement si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commentaire de l'ordre.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode BuyStop(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultOrder().

© 2000-2024, MetaQuotes Ltd.

SellLimit

Place un ordre en attente de type Sell Limit (vente au prix, plus haut que le prix actuel du marché) avec les paramètres spécifiés.

```
bool  SellLimit(
   double                volume,                      // volume de l'ordre
   double                price,                       // prix de l'ordre
   const string          symbol=NULL,                 // symbole
   double                sl=0.0,                      // prix du stop loss
   double                tp=0.0,                      // prix du take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,    // durée de vie de l'ordre
   datetime              expiration=0,                // heure d'expiration de l'ordr
   const string          comment=""                   // commentaire
   )
```

Paramètres

volume

[in] Volume de l'ordre.

price

[in] Prix de l'ordre.

symbol=NULL

[in] Symbole de l'ordre. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

type_time=ORDER_TIME_GTC

[in] Durée de vie de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Heure d'expiration de l'ordre (utilisé uniquement si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commentaire de l'ordre.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode SellLimit(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultOrder().

© 2000-2024, MetaQuotes Ltd.

SellStop

Place un ordre en attente de type Sell Stop (vente au prix, plus bas que le prix actuel du marché) avec les paramètres spécifiés.

```
bool  SellStop(
   double                volume,                      // volume de l'ordre
   double                price,                       // prix de l'ordre
   const string          symbol=NULL,                 // symbole
   double                sl=0.0,                      // prix du stop loss
   double                tp=0.0,                      // prix du take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,    // durée de vie de l'ordre
   datetime              expiration=0,                // heure d'expiration de l'ordr
   const string          comment=""                   // commentaire
   )
```

Paramètres

volume

[in] Volume de l'ordre.

price

[in] Prix de l'ordre.

symbol=NULL

[in] Symbole de l'ordre. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

type_time=ORDER_TIME_GTC

[in] Durée de vie de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Heure d'expiration de l'ordre (utilisé uniquement si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commentaire de l'ordre.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode SellStop(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultOrder().

© 2000-2024, MetaQuotes Ltd.

Request

Retourne une copie de la structure de la dernière requête.

```
void  Request(
   MqlTradeRequest&  request      // structure destination
   ) const
```

Paramètres

request

[out] Référence à une structure de type MqlTradeRequest.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

RequestAction

Retourne le type d'une opération de trading.

```
ENUM_TRADE_REQUEST_ACTIONS  RequestAction() const
```

Valeur de retour

Type de l'opération de trading, utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestActionDescription

Retourne le type d'une opération de trading sous forme d'une chaîne de caractères.

```
string  RequestActionDescription() const
```

Valeur de retour

Type de l'opération de trading (sous forme d'une chaîne de caractères), utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestMagic

Retourne l'identifiant Magic de l'Expert Advisor.

```
ulong  RequestMagic() const
```

Valeur de retour

L'identifiant Magic de l'Expert Advisor, utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestOrder

Retourne le ticket de l'ordre utilisé dans la dernière requête.

```
ulong  RequestOrder() const
```

Valeur de retour

Le ticket de l'ordre de la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestSymbol

Retourne le nom du symbole utilisé dans la dernière requête.

```
string  RequestSymbol() const
```

Valeur de retour

Le nom du symbole utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestVolume

Retourne le volume de trading (en lots) utilisé dans la dernière requête.

```
double  RequestVolume() const
```

Valeur de retour

Le volume de trading (en lots), utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestPrice

Retourne le prix utilisé dans la dernière requête.

```
double  RequestPrice() const
```

Valeur de retour

Le prix de l'ordre utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestStopLimit

Retourne le prix d'un ordre en attente de type Stop Limit, utilisé dans la dernière requête.

```
double  RequestStoplimit() const
```

Valeur de retour

Le prix de l'ordre en attente de type Stop Limit, utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestSL

Retourne le prix du Stop Loss de l'ordre, utilisé dans la dernière requête.

```
double  RequestSL() const
```

Valeur de retour

Le prix du Stop Loss, utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestTP

Retourne le prix du Take Profit de l'ordre, utilisé dans la dernière requête.

```
double  RequestTP() const
```

Valeur de retour

Le prix du Take Profit , utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestDeviation

Retourne la déviation du prix de l'ordre, utilisé dans la dernière requête.

```
ulong  RequestDeviation() const
```

Valeur de retour

La déviation du prix de l'ordre, utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestType

Retourne le type de l'ordre, utilisé dans la dernière requête.

```
ENUM_ORDER_TYPE  RequestType() const
```

Valeur de retour

Le type de l'ordre utilisé dans la dernière requête (valeur de l'énumération ENUM_ORDER_TYPE).

© 2000-2024, MetaQuotes Ltd.

RequestTypeDescription

Retourne le type de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête.

```
string  RequestTypeDescription() const
```

Valeur de retour

Le type de l'ordre (sous forme d'une chaîne de caractères) utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestTypeFilling

Retourne le type de remplissage de l'ordre, utilisé dans la dernière requête.

```
ENUM_ORDER_TYPE_FILLING  RequestTypeFilling() const
```

Valeur de retour

Le type de remplissage de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_FILLING), utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestTypeFillingDescription

Retourne le type de remplissage de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête.

```
string  RequestTypeFillingDescription() const
```

Valeur de retour

Le type de remplissage (sous forme d'une chaîne de caractères) de l'ordre, utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestTypeTime

Retourne la période de validité de l'ordre, utilisé dans la dernière requête.

```
ENUM_ORDER_TYPE_TIME  RequestTypeTime() const
```

Valeur de retour

La période de validité de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME), utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestTypeTimeDescription

Retourne la période de validité de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête.

```
string  RequestTypeTimeDescription() const
```

Valeur de retour

La période de validité de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestExpiration

Retourne l'heure d'expiration de l'ordre utilisé dans la dernière requête.

```
datetime  RequestExpiration() const
```

Valeur de retour

L'heure d'expiration de l'ordre utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestComment

Retourne le commentaire de l'ordre utilisé dans la dernière requête.

```
string  RequestComment() const
```

Valeur de retour

Le commentaire de l'ordre utilisé dans la dernière requête.

© 2000-2024, MetaQuotes Ltd.

RequestPosition

Retourne le ticket d'une position.

```
ulong  RequestPosition() const
```

Valeur de Retour

Le ticket de la position utilisée dans la dernière demande.

© 2000-2024, MetaQuotes Ltd.

RequestPositionBy

Retourne le ticket de la position opposée.

```
ulong  RequestPositionBy() const
```

Valeur de Retour

Le ticket de la position opposée utilisée dans la dernière demande.

© 2000-2024, MetaQuotes Ltd.

Result

Retourne une copie de la structure des résultats de la dernière requête.

```
void  Result(
   MqlTradeResult&  result      // référence
   ) const
```

Paramètres

result

[out] Référence à une structure de type MqlTradeResult.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

ResultRetcode

Retourne le code du résultat de la requête.

```
uint  ResultRetcode() const
```

Valeur de retour

Le Code du résultat de la requête.

© 2000-2024, MetaQuotes Ltd.

ResultRetcodeDescription

Retourne le code du résultat de la requête sous forme de texte.

```
string  ResultRetcodeDescription() const
```

Valeur de retour

Le code du résultat de la dernière requête sous forme de texte.

© 2000-2024, MetaQuotes Ltd.

ResultDeal

Retourne le ticket du deal.

```
ulong  ResultDeal() const
```

Valeur de retour

Le ticket du deal, si le deal est exécuté.

© 2000-2024, MetaQuotes Ltd.

ResultOrder

Retourne le ticket de l'ordre.

```
ulong  ResultOrder() const
```

Valeur de retour

Le ticket de l'ordre, si l'ordre est placé.

© 2000-2024, MetaQuotes Ltd.

ResultVolume

Retourne le volume du deal ou de l'ordre.

```
double ResultVolume() const
```

Valeur de retour

Le volume du deal ou de l'ordre.

© 2000-2024, MetaQuotes Ltd.

ResultPrice

Retourne le prix confirmé par le courtier.

```
double  ResultPrice() const
```

Valeur de retour

Le prix, confirmé par le courtier.

© 2000-2024, MetaQuotes Ltd.

ResultBid

Retourne le prix bid courant (la recotation).

```
double  ResultBid() const
```

Valeur de retour

Le prix bid courant (la recotation).

© 2000-2024, MetaQuotes Ltd.

ResultAsk

Retourne le prix ask courant (la recotation).

```
double  ResultAsk() const
```

Valeur de retour

Le prix ask courant (la recotation).

© 2000-2024, MetaQuotes Ltd.

ResultComment

Retourne le commentaire du courtier.

```
string ResultComment() const
```

Valeur de retour

Le commentaire du courtier pour l'opération.

© 2000-2024, MetaQuotes Ltd.

CheckResult

Retourne une copie de la structure des résultats de la dernière requête.

```
void  CheckResult(
   MqlTradeCheckResult&  check_result      // référence
   ) const
```

Paramètres

check_result

[out] Référence à la structure destination de type MqlTradeCheckResult.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CheckResultRetcode

Retourne la valeur du champ retcode du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
uint  CheckResultRetcode() const
```

Valeur de retour

La valeur du champ retcode (code d'erreur) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2024, MetaQuotes Ltd.

CheckResultRetcodeDescription

Retourne la description textuelle du champ retcode du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
string  ResultRetcodeDescription() const
```

Valeur de retour

La description textuelle du champ retcode (code d'erreur) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2024, MetaQuotes Ltd.

CheckResultBalance

Retourne la valeur du champ balance du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultBalance() const
```

Valeur de retour

La valeur du champ balance (la valeur de la balance qui résultera de l'exécution de l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2024, MetaQuotes Ltd.

CheckResultEquity

Retourne la valeur du champ equity du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultEquity() const
```

Valeur de retour

La valeur du champ equity (la valeur du capital qui résultera de l'exécution de l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2024, MetaQuotes Ltd.

CheckResultProfit

Retourne la valeur du champ profit du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultProfit() const
```

Valeur de retour

La valeur du champ profit (la valeur du profit qui résultera de l'exécution de l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2024, MetaQuotes Ltd.

CheckResultMargin

Retourne la valeur du champ margin du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double CheckResultMargin() const
```

Valeur de retour

La valeur du champ margin (la marge requise pour l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2024, MetaQuotes Ltd.

CheckResultMarginFree

Retourne la valeur du champ margin_free du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultMarginFree() const
```

Valeur de retour

La valeur du champ margin_free (la marge libre qui résultera de l'exécution de l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2024, MetaQuotes Ltd.

CheckResultMarginLevel

Retourne la valeur du champ margin_level du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultMarginLevel() const
```

Valeur de retour

La valeur du champ margin_level (le niveau de marge qui résultera de l'exécution de l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2024, MetaQuotes Ltd.

CheckResultComment

Retourne la valeur du champ comment du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
string  CheckResultComment() const
```

Valeur de retour

La valeur du champ comment (commentaire de code de retour, description de l'erreur) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2024, MetaQuotes Ltd.

PrintRequest

Ecrit les paramètres de la dernière requête dans le journal.

```
void  PrintRequest() const
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

PrintResult

Ecrit les résultats de la dernière requête dans le journal.

```
void PrintResult() const
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

FormatRequest

Prépare la chaîne de caractères formatée avec les paramètres de la dernière requête.

```
string  FormatRequest(
   string&                 str,          // chaîne de caractères destination
   const MqlTradeRequest&  request       // requête
   ) const
```

Paramètres

str

[in] Chaîne de caractères destination, passée par référence.

request

[in] Une structure de type MqlTradeRequest avec les paramètres de la dernière requête.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

FormatRequestResult

Prépare la chaîne de caractères formatée avec les résultats d'exécution de la dernière requête.

```
string  FormatRequestResult(
   string&                 str,          // chaîne de caractères
   const MqlTradeRequest&  request,      // structure de la requête
   const MqlTradeResult&   result        // structure de résultat
   ) const
```

Paramètres

str

[in] Chaîne de caractères destination, passée par référence.

request

[in] Une structure de type MqlTradeRequest avec les paramètres de la dernière requête.

result

[in] Une structure de type MqlTradeResult avec les résultats de la dernière requête.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CTerminalInfo

CString est une classe permettant un accès simplifié aux propriétés de l'environnement du programme MQL5.

Description

La classe CTerminalInfo permet d'accéder aux propriétés de l'environnement du programme MQL5.

Déclaration

```
class CTerminalInfo : public CObject
```

Titre

```
#include <Trade\TerminalInfo.mqh>
```

Hiérarchie d'héritage

CObject
  CTerminalInfo

Méthodes de classe par groupes

Méthodes d'accès aux propriétés de type integer	
Build	Retourne le numéro de build du terminal client
IsConnected	Retourne les informations de connexion au serveur de trading
IsDLLsAllowed	Retourne l'autorisation d'utiliser des DLL
IsTradeAllowed	Retourne l'autorisation d'effectuer des opération de trading
IsEmailEnabled	Retourne l'autorisation d'envoyer des emails au serveur/login SMTP spécifié dans les paramètres du terminal
IsFtpEnabled	Retourne l'autorisation d'envoyer des rapports de trading au serveur/login FTP spécifié dans les paramètres du terminal
MaxBars	Retourne le nombre maximum de barres du graphique
CodePage	Retourne la page de code de la langue du terminal client
CPUCores	Retourne le nombre de coeurs CPU
MemoryPhysical	Retourne la quantité de mémoire physique (en Mo)
MemoryTotal	Retourne la quantité de mémoire totale disponible pour le processus de l'agent ou du terminal (en Mo)
MemoryAvailable	Retourne la quantité de mémoire libre, disponible pour pour le processus de l'agent ou du terminal (en Mo)

© 2000-2024, MetaQuotes Ltd.

Méthodes d'accès aux propriétés de type integer	
MemoryUsed	Retourne la quantité de mémoire utilisée par le processus de l'agent ou du terminal (en Mo)
IsX64	Retourne le type du terminal client (32/64 bits)
OpenCLSupport	Retourne la version de OpenCL supportée par la carte vidéo
DiskSpace	Retourne la quantité d'espace disque disponible (en Mo)
Méthodes d'accès aux propriétés de type string	
Language	Retourne la langue du terminal client
Name	Retourne le nom du terminal client
Company	Retourne la société du terminal client
Path	Retourne le répertoire du terminal client
DataPath	Retourne le répertoire des données du terminal client
CommonDataPath	Retourne le répertoire des données communes de tous les terminaux client installés sur l'ordinateur
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété de type integer
InfoString	Retourne la valeur de la propriété de type string

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Build

Retourne le numéro de build du terminal client.

```
int  CBuild() const
```

Valeur de retour

Le numéro de build du terminal client.

Note

Pour récupérer le numéro de build, la fonction TerminalInfoInteger() est appelée (**propriété** TERMINAL_BUILD).

© 2000-2024, MetaQuotes Ltd.

IsConnected

Retourne les informations de connexion au serveur de trading.

```
bool  IsConnected() const
```

Valeur de retour

vrai si le terminal est connecté au serveur de trading, faux sinon.

Note

Pour récupérer le statut de la connexion, la fonction TerminalInfoInteger() est appelée (**propriété** TERMINAL_CONNECTED).

© 2000-2024, MetaQuotes Ltd.

IsDLLsAllowed

Retourne l'autorisation d'utiliser des DLL.

```
bool  IsDLLsAllowed() const
```

Valeur de retour

vrai si l'utilisation des DLL est autorisée, faux sinon.

Note

Pour récupérer l'autorisation d'utiliser les DLL, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_DLLS_ALLOWED).

© 2000-2024, MetaQuotes Ltd.

IsTradeAllowed

Retourne l'autorisation d'effectuer des opération de trading.

```
bool  IsTradeAllowed() const
```

Valeur de retour

vrai si le trading est autorisé, faux sinon.

Note

Pour récupérer l'autorisation d'effectuer des opérations de trading, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_TRADE_ALLOWED).

© 2000-2024, MetaQuotes Ltd.

IsEmailEnabled

Retourne l'autorisation d'envoyer des emails au serveur/login SMTP spécifié dans les paramètres du terminal.

```
bool  IsEmailEnabled() const
```

Valeur de retour

vrai si l'envoi d'emails est autorisé, faux sinon.

Note

Pour récupérer l'autorisation d'envoyer des emails, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_EMAIL_ENABLED).

© 2000-2024, MetaQuotes Ltd.

IsFtpEnabled

Retourne l'autorisation d'envoyer des rapports de trading au serveur/login FTP spécifié dans les paramètres du terminal.

```
bool  IsFtpEnabled() const
```

Valeur de retour

vrai si l'envoi de rapports de trading au serveur FTP est autorisé, faux sinon.

Note

Pour récupérer l'autorisation d'envoyer des rapports de trading, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_FTP_ENABLED).

© 2000-2024, MetaQuotes Ltd.

MaxBars

Retourne le nombre maximum de barres du graphique spécifié dans les paramètres du terminal client.

```
int  MaxBars() const
```

Valeur de retour

Le nombre maximum de barres du graphique.

Note

Pour récupérer le nombre de barres du graphique, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_MAXBARS).

© 2000-2024, MetaQuotes Ltd.

CodePage

Retourne la page de code de la langue du terminal client.

```
int  CodePage() const
```

Valeur de retour

La page de code de la langue du terminal client.

Note

Pour récupérer la page de code, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_CODEPAGE).

© 2000-2024, MetaQuotes Ltd.

CPUCores

Retourne le nombre de coeurs CPU du système.

```
int  CPUCores() const
```

Valeur de retour

Le nombre de coeur CPU du système.

Note

Pour récupérer le nombre de coeurs CPU, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_CPU_CORES).

© 2000-2024, MetaQuotes Ltd.

MemoryPhysical

Retourne la quantité de mémoire physique (en Mo).

```
int  MemoryPhysical() const
```

Valeur de retour

La quantité de mémoire physique (en Mo).

Note

Pour récupérer la mémoire physique, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_MEMORY_PHYSICAL).

© 2000-2024, MetaQuotes Ltd.

MemoryTotal

Retourne la quantité de mémoire totale, disponible pour le terminal client ou l'agent (en Mo).

```
int  MemoryTotal() const
```

Valeur de retour

La quantité de mémoire totale (en Mo) disponible pour le terminal ou le client.

Note

Pour récupérer la mémoire totale, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_MEMORY_TOTAL).

© 2000-2024, MetaQuotes Ltd.

MemoryAvailable

Retourne la quantité de mémoire libre, disponible pour pour le terminal client ou l'agent (en Mo).

```
int  MemoryTotal() const
```

Valeur de retour

La quantité de mémoire libre (en Mo), disponible pour pour le terminal client ou l'agent.

Note

Pour récupérer la mémoire libre, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_MEMORY_TOTAL).

© 2000-2024, MetaQuotes Ltd.

MemoryUsed

Retourne la quantité de mémoire utilisée par le terminal client ou l'agent (en Mo).

```
int  MemoryUsed() const
```

Valeur de retour

La quantité de mémoire utilisée par le terminal ou l'agent (en Mo).

Note

Pour récupérer la quantité de mémoire, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_MEMORY_USED).

© 2000-2024, MetaQuotes Ltd.

IsX64

Retourne le type du terminal client (32/64 bits).

```
bool  IsX64() const
```

Valeur de retour

vrai si la version 64 bits est utilisée, faux sinon.

Note

Pour récupérer le type du terminal client, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_X64).

© 2000-2024, MetaQuotes Ltd.

OpenCLSupport

Retourne la version de OpenCL supportée par la carte vidéo.

```
int  OpenCLSupport() const
```

Valeur de retour

La valeur retournée est de la forme : 0x00010002 = "1,2". 0 signifie que OpenCL n'est pas supporté.

Note

Pour récupérer la version d'OpenCL, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_OPENCL_SUPPORT).

© 2000-2024, MetaQuotes Ltd.

DiskSpace

Retourne la quantité d'espace disque disponible pour le terminal ou l'agent (en Mo).

```
int  MDiskSpace() const
```

Valeur de retour

Quantité d'espace disque disponible (en Mo), disponible pour le terminal client ou l'agent (pour les fichiers sauvegardés dans le répertoire MQL5\Files).

Note

Pour récupérer l'espace disque disponible, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_DISK_SPACE).

© 2000-2024, MetaQuotes Ltd.

Language

Retourne la langue du terminal client.

```
string Language() const
```

Valeur de retour

La langue utilisée dans le terminal client.

Note

Pour récupérer la langue, la fonction TerminalInfoString() est appelée (propriété TERMINAL_LANGUAGE).

© 2000-2024, MetaQuotes Ltd.

Name

Retourne le nom du terminal client.

```
string  Name() const
```

Valeur de retour

Le nom du terminal client.

Note

Pour récupérer le nom du terminal client, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_NAME).

© 2000-2024, MetaQuotes Ltd.

Company

Retourne le nom du courtier.

```
string  Company() const
```

Valeur de retour

Le nom du courtier.

Note

Pour récupérer le nom du courtier, la fonction TerminalInfoString() est appelée (propriété TERMINAL_COMPANY).

© 2000-2024, MetaQuotes Ltd.

Path

Retourne le répertoire du terminial client.

```
string  Path() const
```

Valeur de retour

Le répertoire du terminal client.

Note

Pour récupérer le répertoire du terminal client, la fonction TerminalInfoString() est appelée (propriété TERMINAL_PATH).

© 2000-2024, MetaQuotes Ltd.

DataPath

Retourne le répertoire de données (data folder) du terminal.

```
string  DataPath() const
```

Valeur de retour

Le répertoire des données du terminal client.

Note

Pour récupérer le répertoire des données du terminal, la fonction TerminalInfoString() est appelée (propriété TERMINAL_DATA_PATH).

© 2000-2024, MetaQuotes Ltd.

CommonDataPath

Retourne le répertoire des données communes de tous les terminaux client installés sur l'ordinateur.

```
string CommonDataPath() const
```

Valeur de retour

Le répertoire des données communes.

Note

Pour récupérer le répertoire des données communes, la fonction TerminalInfoString() est appelée (propriété COMMON_DATA_PATH).

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Retourne la valeur de la propriété correspondante de l'environnement du programme MQL5.

```
int  TerminalInfoInteger(
   int  property_id      // identifiant d'une propriété
   );
```

Paramètres

property_id

[in] Identifiant d'une propriété. Peut être une des valeurs de l'énumération ENUM_TERMINAL_INFO_INTEGER.

Valeur de retour

La valeur de type int.

Note

Ppur récupérer la valeur de la propriété, la fonction TerminalInfoInteger() est appelée.

© 2000-2024, MetaQuotes Ltd.

InfoString

Retourne la valeur de la propriété correspondante de l'environnement du programme MQL5. La propriété doit être de type string.

```
string  TerminalInfoString(
   int  property_id      // identifiant d'une propriété
   );
```

Paramètres

property_id

[in] Identifiant d'une propriété. Peut être une des valeurs de l'énumération ENUM_TERMINAL_INFO_STRING.

Valeur de retour

La valeur de type string.

Note

Ppur récupérer la valeur de la propriété, la fonction TerminalInfoString() est appelée.

© 2000-2024, MetaQuotes Ltd.

Classes de Stratégies de Trading

Cette section contient les détails techniques d'utilisation des classes de création et de test des stratégies de trading et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création de vos stratégies de trading.

La Bibliothèque Standard MQL5 (en terme de stratégies de trading) est située dans le répertoire de travail du terminal dans le répertoire Include\Expert.

Classes de base	Description
CExpertBase	Classe de base pour toutes les classes de stratégies de trading
CExpert	Classe de base pour les Expert Advisor
CExpertSignal	Classe de base pour les classes de Signaux de Trading
CExpertTrailing	Classe de base pour les classes de Trailing Stop (stops suiveurs)
CExpertMoney	Classe de base pour les classes de Money Management

Classes de signal de trading	Description
CSignalAC	Le module des signaux basés sur les modèles de marché de l'indicateur Accelerator Oscillator.
CSignalAMA	Le module des signaux basés sur les modèles de marché de l'indicateur Adaptive Moving Average.
CSignalAO	Le module des signaux basés sur les modèles de marché de l'indicateur Awesome Oscillator.
CSignalBearsPower	Le module des signaux basés sur les modèles de marché de l'oscillateur Bears Power.
CSignalBullsPower	Le module des signaux basés sur les modèles de marché de l'oscillateur Bulls Power.
CSignalCCI	Le module des signaux basés sur les modèles de marché de l'oscillateur Commodity Channel Index.
CSignalDeM	Le module des signaux basés sur les modèles de marché de l'oscillateur DeMarker.
CSignalDEMA	Le module des signaux basés sur les modèles de marché de l'indicateur Double Exponential Moving Average.
CSignalEnvelopes	Le module des signaux basés sur les modèles de marché de l'indicateur Envelopes.
CSignalFrAMA	Le module des signaux basés sur les modèles de marché de l'indicateur Fractal Adaptive Moving Average.

© 2000-2024, MetaQuotes Ltd.

Classes de signal de trading	Description
CSignalITF	Le module de filtration des signaux par date/heure.
CSignalMACD	Le module des signaux basés sur les modèles de marché de l'oscillateur MACD.
CSignalMA	Le module des signaux basés sur les modèles de marché de l'indicateur Moving Average.
CSignalSAR	Le module des signaux basés sur les modèles de marché de l'indicateur Parabolic SAR.
CSignalRSI	Le module des signaux basés sur les modèles de marché de l'oscillateur Relative Strength Index.
CSignalRVI	Le module des signaux basés sur les modèles de marché de l'oscillateur Relative Vigor Index.
CSignalStoch	Le module des signaux basés sur les modèles de marché de l'oscillateur Stochastic.
CSignalTRIX	Le module des signaux basés sur les modèles de marché de l'oscillateur Triple Exponential Average.
CSignalTEMA	Le module des signaux basés sur les modèles de marché de l'indicateur Triple Exponential Moving Average.
CSignalWPR	Le module des signaux basés sur les modèles de marché de l'oscillateur Williams Percent Range.

Classes de Trailing Stop (Stops Suiveurs)	Description
CTrailingFixedPips	Cette classe implémente l'algorithme du Stop Suiveur basé sur des points fixes
CTrailingMA	Cette classe implémente l'algorithme du Stop Suiveur basé sur les valeurs d'une Moyenne Mobile
CTrailingNone	Classe de type stub, elle n'utilise aucun algorithme de Stop Suiveur
CTrailingPSAR	Cette classe implémente l'algorithme du Stop Suiveur basé sur les valeurs de l'indicateur Parabolic SAR

Classes de Money Management	Description
CMoneyFixedLot	Classe utilisant un algorithme basé sur le trading avec une taille de lot fixe.
CMoneyFixedMargin	Classe utilisant un algorithme basé sur le trading avec une marge fixe prédéfinie.

© 2000-2024, MetaQuotes Ltd.

Classes de Money Management	Description
CMoneyFixedRisk	Classe utilisant un algorithme basé sur le trading avec un risque prédéfini.
CMoneyNone	Classe utilisant un algorithme basé sur le trading avec une taille de lot minimum permis.
CMoneySizeOptimized	Classe utilisant un algorithme basé sur le trading avec une taille de lot variable, suivant les résultats des deals précédents.

© 2000-2024, MetaQuotes Ltd.

Classes de base des Expert Advisors

Cette section contient les détails techniques d'utilisation des classes de création et de test des stratégies de trading et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création de vos stratégies de trading.

La Bibliothèque Standard MQL5 (en terme de stratégies de trading) est située dans le répertoire de travail du terminal dans le répertoire Include\Expert.

Classe	Description
CExpertBase	Classe de base pour toutes les classes de stratégies de trading
CExpert	Classe de base pour les Expert Advisor
CExpertSignal	Classe de base pour les classes de Signaux de Trading
CExpertTrailing	Classe de base pour les classes de Trailing Stop (stops suiveurs)
CExpertMoney	Classe de base pour les classes de Money Management

© 2000-2024, MetaQuotes Ltd.

CExpertBase

CExpertBase est la classe de base de la classe CExpert et de toutes les classes de stratégie de trading.

Description

CExpertBase fournit les données et méthodes qui sont communes à tous les objets de l'Expert Advisor.

Déclaration

```
class CExpertBase : public CObject
```

Titre

```
#include <Expert\ExpertBase.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase

Descendants directs

CExpert, CExpertMoney, CExpertSignal, CExpertTrailing

Méthodes de Classe

Méthodes Publiques :

Initialisation	
virtual Init	Méthode d'initialisation de l'instance de classe
virtual ValidationSettings	Vérifie les paramètres
Paramètres	
Symbol	Définit le symbole
Period	Définit la période
Magic	Définit l'identifiant de l'Expert Advisor
Indicateurs et Séries de Données	
virtual SetPriceSeries	sDéfinit les pointeurs vers les séries de données externes (séries de prix)
virtual SetOtherSeries	Définit les pointeurs vers les séries de données externes (séries de données autres que les prix)
virtual InitIndicators	Initialise les indicators et les séries de données
Accés aux Données Protégées	

© 2000-2024, MetaQuotes Ltd.

Initialisation	
InitPhase	Retourne la phase actuelle de l'initialisation de l'objet
TrendType	Définit le type de tendance
UsedSeries	Retourne le masque de bits des séries de données utilisées
EveryTick	Définit le flag "Chaque tick"
Accés au Séries de Données	
Open	Retourne l'élément de la série Open (prix d'ouverture) par son index
High	Retourne l'élément de la série High (plus haut) par son index
Low	Retourne l'élément de la série Low (plus bas) par son index
Close	Retourne l'élément de la série Close (prix de clôture) par son index
Spread	Retourne l'élément de la série Spread par son index
Time	Retourne l'élément de la série Time par son index
TickVolume	Retourne l'élément de la série TickVolume par son index
RealVolume	Retourne l'élément de la série RealVolume (volume réel) par son index

Méthodes Protégées :

Initialisation des Séries de Données	
InitOpen	Méthode d'initialisation de la série Open (ouverture)
InitHigh	Méthode d'initialisation de la série High (plus haut)
InitLow	Méthode d'initialisation de la série Low (plus bas)
InitClose	Méthode d'initialisation de la série Close (prix de clôture)
InitSpread	Méthode d'initialisation de la série Spread
InitTime	Méthode d'initialisation de la série Time
InitTickVolume	Méthode d'initialisation de la série TickVolume
InitRealVolume	Méthode d'initialisation de la série RealVolume
Méthodes Services	
virtual PriceLevelUnit	Retourne l'unité du niveau de prix
virtual StartIndex	Retourne l'index de la première barre à analyser
virtual CompareMagic	Compare l'identifiant de l'Expert Advisor avec la valeur spécifiée.

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

InitPhase

Retourne la phase actuelle de l'initialisation de l'objet.

```
ENUM_INIT_PHASE  InitPhase()
```

Valeur de retour

Phase courant de l'initialisation de l'objet.

Note

L'initialisation de l'objet est faite de plusieurs phases :

1. Démarrage de l'initialisation.

- lancement - après le constructeur
- fin - après la fin avec succès de la méthode Init(...).
- permise - appel de la méthode Init(...)
- non permise - appel de la méthode ValidationSettings() et autres méthodes d'initialisation

2. Phase d'initialisation des paramètres. Durant cette phase, vous devez définir tous les paramètres des objets utilisés pour la création d'indicateurs.

- lancement - après la fin avec succès de la méthode Init(...)
- fin - après la fin avec succès de la méthode ValidationSettings()
- permise - appel des méthodes Symbol(...) et Period(...)
- non permise - appel des méthodes Init(...), SetPriceSeries(...), SetOtherSeries(...) et InitIndicators(...)

3. Vérification des paramètres.

- lancement - après la fin avec succès de la méthode ValidationSettings()
- fin - après la fin avec succès de la méthode InitIndicators(...)
- permise - appel des méthodes Symbol(...), Period(...) et InitIndicators(...)
- non permise - appel de toute autre méthode d'initialisation

4. Fin de l'initialisation.

- lancement - après la fin avec succès de la méthode InitIndicators(...)
- non permise - appel des méthodes d'initialisation

© 2000-2024, MetaQuotes Ltd.

TrendType

Définit le type de tendance.

```
void  TrendType(
   M_TYPE_TREND    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du type de tendance.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

UsedSeries

Retourne le masque de bits des séries de données utilisées.

```
int  UsedSeries()
```

Valeur de retour

La liste des séries de données utilisées sous forme d'un masque de bits.

Note

Si le bit est défini, la série de données correspondante est utilisée, s'il n'est pas défini, la série de données n'est pas utilisée.

Correspondance des bits des séries de données :

bit 0 - série de données Open,
bit 1 - série de données High,
bit 2 - série de données Low,
bit 3 - série de données Close,
bit 4 - série de données Spread,
bit 5 - série de données Time,
bit 6 - série de données TickVolume,
bit 7 - série de données RealVolume.

© 2000-2024, MetaQuotes Ltd.

EveryTick

Définit le flag "Chaque tick".

```
void  EveryTick(
   bool    value         // flag
   )
```

Paramètres

value

[in] Nouvelle valeur du flag.

Valeur de retour

Aucune.

Note

Si le flag n'est pas défini, la méthode de traitement n'est appelée que sur la première barre de la période et du symbole courants.

© 2000-2024, MetaQuotes Ltd.

Open

Retourne l'élément de la série Open (prix d'ouverture) par son index.

```
double  Open(
   int    ind         // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série Open situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2024, MetaQuotes Ltd.

High

Retourne l'élément de la série High par son index.

```
double  High(
   int    ind         // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série High situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2024, MetaQuotes Ltd.

Low

Retourne l'élément de la série Low (plus bas) par son index.

```
double  Low(
   int    ind          // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série Low situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2024, MetaQuotes Ltd.

Close

Retourne l'élément de la série Close par son index.

```
double  Close(
   int    ind         // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série Close situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2024, MetaQuotes Ltd.

Spread

Retourne l'élément de la série Spread par son index.

```
double  Spread(
   int    ind         // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série Spread situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2024, MetaQuotes Ltd.

Time

Retourne l'élément de la série Time par son index

```
datetime  Time(
   int    ind          // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série Time situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2024, MetaQuotes Ltd.

TickVolume

Retourne l'élément de la série TickVolume par son index.

```
long  TickVolume(
   int    ind         // Index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série TickVolume situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2024, MetaQuotes Ltd.

RealVolume

Retourne l'élément de la série RealVolume (volume réel) par son index.

```
long  RealVolume(
   int    ind         // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série de données RealVolume situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2024, MetaQuotes Ltd.

Init

Initialise l'objet.

```
bool  Init(
   CSymbolInfo       symbol,    // symbole
   ENUM_TIMEFRAMES   period,    // période
   double            point      // point
   )
```

Paramètres

symbol

[in] Pointeur sur l'objet de type CSymbolInfo pour accéder aux informations du symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

point

[in] Le "poids" du point de 2/4 chiffres.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Symbol

Définit le symbole.

```
bool  Symbol(
   string    name     // symbole
   )
```

Paramètres

name

[in] Symbole.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La définition du symbole est nécessaire si l'objet utilise un symbole différent de celui défini au moment de l'initialisation.

© 2000-2024, MetaQuotes Ltd.

Period

Définit la période.

```
bool  Period(
   ENUM_TIMEFRAMES  value     // période
   )
```

Paramètres

value

[in] Période.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La définition de la période est nécessaire si l'objet utilise une période différente de la période définie au moment de l'initialisation.

© 2000-2024, MetaQuotes Ltd.

Magic

Définit l'identifiant de l'Expert Advisor.

```
void  Magic(
   ulong  value     // magic
   )
```

Paramètres

value

[in] Identifiant de l'Expert Advisor.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

SetPriceSeries

Définit les pointeurs vers les séries de données externes (séries de données autres que les prix).

```
virtual bool  SetPriceSeries(
   CiOpen*    open,         // pointeur
   CiHigh*    high,         // pointeur
   CiLow*     low,          // pointeur
   CiClose*   close         // pointeur
   )
```

Paramètres

open

[in] Pointeur sur la série de données Open (prix d'ouverture).

high

[in] Pointeur sur la série de données High (plus haut).

low

[in] Pointeur sur la série de données Low (plus bas).

close

[in] Pointeur sur la série de données Close (prix de clôture).

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La définition des pointeurs sur les séries de données externes (séries de données des prix) est nécessaire si l'objet utilise des séries de données d'un symbole ou d'une période différents de ceux définis au moment de l'initialisation.

© 2000-2024, MetaQuotes Ltd.

SetOtherSeries

Définit les pointeurs vers les séries de données externes (séries de données autres que les prix).

```
virtual bool  SetOtherSeries(
   CiSpread*       spread,        // pointeur
   CiTime*         time,          // pointeur
   CiTickVolume*   tick_volume,   // pointeur
   CiRealVolume*   real_volume    // pointeur
   )
```

Paramètres

spread

[in] Pointeur sur la série de données Spread.

time

[in] Pointeur sur la série de données Time.

tick_volume

[in] Pointeur sur la série de données TickVolume.

real_volume

[in] Pointeur sur la série de données RealVolume.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La définition des pointeurs sur les séries de données externes (séries de données autres que les prix) est nécessaire si l'objet utilise des séries de données d'un symbole ou d'une période différents de ceux définis au moment de l'initialisation.

© 2000-2024, MetaQuotes Ltd.

InitIndicators

Initialise tous les indicateurs et les séries de données.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Les séries de données ne sont initialisées que si l'objet utilise le symbole ou la période définit lors de l'initialisation et que le symbole ou la période sont différents.

© 2000-2024, MetaQuotes Ltd.

InitOpen

Initialise la série de données Open (prix d'ouverture).

```
bool  InitOpen(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données Open n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2024, MetaQuotes Ltd.

InitHigh

Initialise la série de données High (plus haut).

```
bool  InitHigh(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données High n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2024, MetaQuotes Ltd.

InitLow

Initialise la série de données Low (plus bas).

```
bool  InitLow(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données Low n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2024, MetaQuotes Ltd.

InitClose

Initialise la série de données Close (prix de clôture).

```
bool  InitClose(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données Clôture n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2024, MetaQuotes Ltd.

InitSpread

Initialise la série de données Spread.

```
bool  InitSpread(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données Spread n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2024, MetaQuotes Ltd.

InitTime

Initialise la série de données Time (date/heure).

```
bool  InitTime(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données Time n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2024, MetaQuotes Ltd.

InitTickVolume

Initialise la série de données TickVolume.

```
bool  InitTickVolume(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données TickVolume n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2024, MetaQuotes Ltd.

InitRealVolume

Initialise la série de données RealVolume (volume réel).

```
bool  InitRealVolume(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données RealVolume n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2024, MetaQuotes Ltd.

PriceLevelUnit

Retourne l'unité du niveau de prix.

```
virtual double  PriceLevelUnit()
```

Valeur de retour

Valeur de l'unité du niveau de prix.

Note

La méthode de la classe de base retourne le "poids" du point à 2/4 décimales.

© 2000-2024, MetaQuotes Ltd.

StartIndex

Retourne l'index de la première barre à analyser.

```
virtual int  StartIndex()
```

Valeur de retour

L'index de la première barre à analyser.

Note

La méthode retourne 0 si le flag pour analyser la barre courante est défini à vrai (analyse depuis la barre courante). Si le flag n'est pas défini, retourne 1 (analyse depuis la dernière barre complétée).

© 2000-2024, MetaQuotes Ltd.

CompareMagic

Compare l'identifiant de l'Expert Advisor avec la valeur spécifiée.

```
virtual bool  CompareMagic(
   ulong  magic    // Valeur de comparaison
   )
```

Paramètres

magic

[in] Valeur de comparaison.

Valeur de retour

vrai s'ils sont égaux, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CExpert

Classe de base pour toutes les classes de stratégies de trading. Des algorithmes sont intégrés permettant de travailler avec les séries de données et les indicateurs, ainsi qu'un ensemble de méthodes virtuelles pour les stratégies de trading.

Comment l'utiliser :

1. Préparez l'algorithme de la stratégie ;
2. Creéez votre propre classe, dérivant de la classe CExpert ;
3. Surchargez les méthodes virtuelles de votre classe avec vos propres algorithmes.

Description

La classe CExpert est un ensemble de méthodes virtuelles pour l'implémentation des stratégies de trading.

Déclaration

```
class CExpert : public CExpertBase
```

Titre

```
#include <Expert\Expert.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpert

Méthodes de Classe

Initialisation	
Init	Méthode d'initialisation de l'instance de classe
virtual InitSignal	Initialise l'objet Signal de Trading
virtual InitTrailing	Initialise l'objet Stop Suiveur (Trailing Stop)
virtual InitMoney	Initialise l'objet Money Management
virtual InitTrade	Initialise l'objet Trade
virtual ValidationSettings	Vérifie les paramètres
virtual InitIndicators	Initialise les indicateurs et les séries de données
virtual InitParameters	Méthode d'initialisation des paramètres
virtual Deinit	Méthode de dé-initialisation de l'instance de classe
virtual DeinitSignal	Dé-initialise l'objet Signal de Trading

© 2000-2024, MetaQuotes Ltd.

Initialisation	
virtual DeinitTrailing	Dé-initialise l'objet Stop Suiveur (Trailing Stop)
virtual DeinitMoney	Dé-initialise l'objet Money Management
virtual DeinitTrade	Dé-initialise l'objet Trade
virtual DeinitIndicators	Dé-initialise les indicateurs et les séries de données
Paramètres	
Magic	Définit l'identifiant de l'Expert Advisor
MaxOrders	Retourne/Définit le nombre maximum d'ordres autorisés
OnTickProcess	Définit un flag pour traiter l'évènement "OnTick"
OnTradeProcess	Définit un flag pour traiter l'évènement "OnTrade"
OnTimerProcess	Définit un flag pour traiter l'évènement "OnTimer"
OnChartEventProcess	Définit un flag pour traiter l'évènement "OnChartEvent"
OnBookEventProcess	Définit un flag pour traiter l'évènement "OnBookEvent"
Méthodes de Traitement des Evènements	
OnTick	Gestionnaire d'évènement OnTick
OnTrade	Gestionnaire d'évènement OnTrade
OnTimer	Gestionnaire d'évènement OnTimer
OnChartEvent	Gestionnaire d'évènement OnChartEvent
OnBookEvent	Gestionnaire d'évènement OnBookEvent
Méthodes de Mise à Jour	
Refresh	Met à jour toutes les données
Traitement	
Traitement	Algorithme principal de traitement
Méthodes d'Entrée sur le Marché	
CheckOpen	Vérifie les conditions d'ouverture d'une position
CheckOpenLong	Vérifie les conditions d'ouverture d'une position longue
CheckOpenShort	Vérifie les conditions d'ouverture d'une position courte (short)
OpenLong	Ouvre une position longue
OpenShort	Ouvre une position courte (short)
Méthodes de Sortie du Marché	

© 2000-2024, MetaQuotes Ltd.

Initialisation	
CheckClose	Vérifie les conditions pour clôturer la position courante
CheckCloseLong	Vérifie les conditions pour clôturer une position longue
CheckCloseShort	Vérifie les conditions pour clôturer une position courte (short)
CloseAll	Ferme toutes les positions ouvertes et supprime tous les ordres
Close	Ferme la position ouverte
CloseLong	Ferme la position longue
CloseShort	Ferme la position courte (short)
Méthodes de Renversement de Position	
CheckReverse	Vérifie les conditions pour renverser une position ouverte
CheckReverseLong	Vérifie les conditions pour renverser une position longue
CheckReverseShort	Vérifie les conditions pour renverser une position courte (short)
ReverseLong	Effectue le renversement d'une position longue
ReverseShort	Effectue le renversement d'une position courte (short)
Méthodes de Suivi d'une Position/ d'un Ordre	
CheckTrailingStop	Vérifie les conditions pour modifier les paramètres d'une position
CheckTrailingStopLong	Vérifie les conditions du Stop Suiveur (Trailing Stop) d'une position longue
CheckTrailingStopShort	Vérifie les conditions du Stop Suiveur (Trailing Stop) d'une position courte (short)
TrailingStopLong	Positionne le Stop Suiveur (Trailing Stop) d'une position longue
TrailingStopShort	Positionne le Stop Suiveur (Trailing Stop) d'une position courte (short)
CheckTrailingOrderLong	Vérifie les conditions du Stop Suiveur (Trailing Stop) d'un ordre en attente de type Buy Limit/Stop
CheckTrailingOrderShort	Vérifie les conditions du Stop Suiveur (Trailing Stop) d'un ordre en attente de type Sell Limit/Stop
TrailingOrderLong	Positionne le Stop Suiveur (Trailing Stop) d'un ordre en attente de type Buy Limit/Stop

© 2000-2024, MetaQuotes Ltd.

Initialisation	
TrailingOrderShort	Positionne le Stop Suiveur (Trailing Stop) d'un ordre en attente de type Sell Limit/Stop
Méthodes de Suppression d'un Ordre	
CheckDeleteOrderLong	Vérifie les conditions pour supprimer un ordre en attente de type Buy
CheckDeleteOrderShort	Vérifie les conditions pour supprimer un ordre en attente de type Sell
DeleteOrders	Supprime tous les ordres
DeleteOrder	Supprime un ordre en attente de type Stop/Limit
DeleteOrderLong	Supprime un ordre en attente de type Buy Limit/Stop
DeleteOrderShort	Supprime un ordre en attente de type Sell Limit/Stop
Méthodes liées aux Volumes de Trading	
LotOpenLong	Retourne le volume de trading pour une opération de type buy
LotOpenShort	Retourne le volume de trading pour une opération de type sell
LotReverse	Retourne le volume de trading pour une opération de renversement d'une position
Méthodes d'Historique de Trading	
PrepareHistoryDate	Définit la date de début de suivi de l'historique de trading
HistoryPoint	Crée un point de contrôle de l'historique de trading (sauvegarde le nombre de positions, les ordres, les deals et les ordres historiques)
CheckTradeState	Compare l'état courant avec celui sauvegardé et appelle le gestionnaire d'évènement correspondant
Flags d'Evènement	
WaitEvent	Définit le flag d'attente d'évènement
NoWaitEvent	Réinitialise le flag d'attente d'évènement
Méthodes de Traitement des Evènements de Trading	
TradeEventPositionStopTake	Gestionnaire de l'évènement "Stop Loss/Take Profit d'une position déclenché"
TradeEventOrderTriggered	Gestionnaire de l'évènement "Ordre en Attente Déclenché"
TradeEventPositionOpened	Gestionnaire de l'évènement "Position Ouverte"

© 2000-2024, MetaQuotes Ltd.

Initialisation	
TradeEventPositionVolumeChanged	Gestionnaire de l'évènement "Volume de la Position Modifié"
TradeEventPositionModified	Gestionnaire de l'évènement "Position Modifiée"
TradeEventPositionClosed	Gestionnaire de l'évènement "Position Fermée"
TradeEventOrderPlaced	Gestionnaire de l'évènement "Ordre en Attente Placé"
TradeEventOrderModified	Gestionnaire de l'évènement "Ordre en Attente Modifié"
TradeEventOrderDeleted	Gestionnaire de l'évènement "Ordre en Attente Supprimé"
TradeEventNotIdentified	Gestionnaire d'un évènement non identifié
Méthodes de Service	
TimeframeAdd	Ajoute une période à suivre
TimeframesFlags	Ensemble des flags des périodes suivies ou pas
SelectPosition	Sélectionne une position pour travailler avec

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2024, MetaQuotes Ltd.

Init

Méthode d'initialisation d'une instance de classe.

```
bool  Init(
   string              symbol,         // symbole
   ENUM_TIMEFRAMES     period,         // période
   bool                every_tick,     // flag
   ulong               magic           // magic
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

every_tick

[in] Flag.

magic

[in] Identifiant de l'Expert Advisor (numéro Magic).

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si every_tick est à true, la méthode Processing() est appelée à chaque tick du symbole en cours. Sinon, la méthode Processing() n'est appelée qu'à l'apparition d'une nouvelle barre du symbole en cours.

© 2000-2024, MetaQuotes Ltd.

Magic

Définit l'identifiant (magic) de l'Expert Advisor.

```
void  Magic(
   ulong  value     // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de l'identifiant de l'Expert Advisor.

Valeur de retour

Aucune.

Note

Cette fonction définit la valeur de l'identifiant (magic) de l'Expert Advisor aux classes suivantes : Trade, Signal, Money, Trailing.

Implémentation

```
//+------------------------------------------------------------------+
//| Définit le nombre magic à l'objet et ses dépendances. |
//| ENTREE :  value - nouvelle valeur du nombre magic. |
//| SORTIE : aucune.                                                 |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
void CExpert::Magic(ulong value)
  {
   if(m_trade!=NULL)     m_trade.SetExpertMagicNumber(value);
   if(m_signal!=NULL)    m_signal.Magic(value);
   if(m_money!=NULL)     m_money.Magic(value);
   if(m_trailing!=NULL) m_trailing.Magic(value);
//---
   CExpertBase::Magic(value);
  }
```

© 2000-2024, MetaQuotes Ltd.

InitSignal

Initialise l'objet Signal de Trading.

```
virtual bool  InitSignal(
   CExpertSignal*     signal=NULL,         // pointeur
   )
```

Paramètres

signal

[in] Pointeur sur l'objet de classe CExpertSignal (ou une classe dérivée).

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si le paramètre signal est NULL, la classe CExpertSignal sera utilisée, elle ne fait rien.

© 2000-2024, MetaQuotes Ltd.

InitTrailing

Initialise l'objet Stop Suiveur (Trailing Stop).

```
virtual bool  InitTrailing(
   CExpertTrailing*     trailing=NULL,        // pointeur
   )
```

Paramètres

trailing

[in] Pointeur sur un objet de classe CExpertTrailing (ou une classe dérivée).

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si le paramètre trailing est NULL, la classe ExpertTrailing sera utilisée, elle ne fait rien.

© 2000-2024, MetaQuotes Ltd.

InitMoney

Initialise l'objet Money Management.

```
virtual bool  InitMoney(
   CExpertMoney*     money=NULL,         // pointeur
   )
```

Paramètres

money

[in] Pointeur sur un objet de classe CExpertMoney (ou un de ses descendants).

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si money est NULL, la classe CExpertMoney sera utilisée, elle utilise le lot minimum.

© 2000-2024, MetaQuotes Ltd.

InitTrade

Initialise l'objet Trade.

```
virtual bool  InitTrade(
   ulong            magic,        // magic
   CExpertTrade*    trade=NULL    // pointeur
   )
```

Paramètres

magic

[in] Identifiant de l'Expert Advisor (sera utilisé dans les demandes de deals).

trade

[in] Pointeur sur un objet CExpertTrade.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Deinit

Méthode de dé-initialisation de l'instance de classe.

```
virtual void Deinit()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnTickProcess

Définit un flag pour traiter l'évènement OnTick.

```
void  OnTickOProcess(
   bool     value        // flag
   )
```

Paramètres

value

[in] Flag pour traiter l'évènement OnTick .

Valeur de retour

Aucune.

Note

Si le flag est à true, l'évènement OnTick est traité, par défaut, le flag est à true.

© 2000-2024, MetaQuotes Ltd.

OnTradeProcess

Définit un flag pour traiter l'évènement OnTrade .

```
void  OnTradeProcess(
   bool     value        // flag
   )
```

Paramètres

value

[in] Flag pour traiter l'évènement OnTrade.

Valeur de retour

Aucune.

Note

Si le flag est à true, l'évènement OnTrade est traité, par défaut, le flag est à false.

© 2000-2024, MetaQuotes Ltd.

OnTimerProcess

Définit un flag pour traiter l'évènement OnTimer.

```
void  OnTimerProcess(
   bool     value        // flag
   )
```

Paramètres

value

[in] Flag pour traiter l'évènement OnTimer.

Valeur de retour

Aucune.

Note

Si le flag est à true, l'évènement OnTimer est traité, par défaut, le flag est à false.

© 2000-2024, MetaQuotes Ltd.

OnChartEventProcess

Définit un flag pour traiter l'évènement OnChartEvent.

```
void  OnChartEventProcess(
   bool     value        // flag
   )
```

Paramètres

value

[in] Flag pour traiter l'évènement OnChartEvent.

Valeur de retour

Aucune.

Note

Si le flag est à true, l'évènement OnChartEvent est traité, par défaut, le flag est à false.

© 2000-2024, MetaQuotes Ltd.

OnBookEventProcess

Définit un flag pour traiter l'évènement Définit un flag pour traiter l'évènement OnBookEvent.

```
void  OnChartEventProcess(
   bool     value        // flag
   )
```

Paramètres

value

[in] Flag pour traiter l'évènement OnBookEvent .

Valeur de retour

Aucune.

Note

Si le flag est à true, l'évènement OnBookEvent est traité, par défaut, le flag est à false.

© 2000-2024, MetaQuotes Ltd.

MaxOrders (Méthode "Get")

Retourne le nombre maximum d'ordres autorisés.

```
int  MaxOrders()
```

Valeur de retour

Nombre maximum d'ordres autorisés.

MaxOrders (Méthode "Set")

Définit le nombre maximum d'ordres autorisés.

```
void  MaxOrders(
   int     max_orders        // Nouvelle valeur
   )
```

Paramètres

max_orders

[in] Nouveau nombre maximum d'ordres autorisés.

Valeur de retour

Aucune.

Note

Par défaut, le nombre maximum d'ordres autorisés est 1.

© 2000-2024, MetaQuotes Ltd.

Signal

Retourne le pointeur sur l'objet Signal de Trading.

```
CExpertSignal*  Signal() const
```

Valeur de retour

Pointeur sur l'objet Signal de Trading.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Vérifie aussi les paramètres de tous les objets de l'Expert Advisor.

© 2000-2024, MetaQuotes Ltd.

InitIndicators

Initialise tous les indicateurs et les séries de données.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Les séries de données sont initialisées si l'objet utilise le symbole ou la période définit lors de l'initialisation et que le symbole ou la période sont différents. La fonction appelle alors la méthode virtuelle InitIndicators() des objets signal de trading, stop suiveur et money management.

© 2000-2024, MetaQuotes Ltd.

OnTick

Gestionnaire de l'évènement OnTick.

```
virtual void  OnTick()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnTrade

Gestionnaire de l'évènement OnTrade.

```
virtual void  OnTrade()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnTimer

Gestionnaire de l'évènement OnTimer.

```
virtual void  OnTimer()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnChartEvent

Gestionnaire de l'évènement OnChartEvent.

```
virtual void  OnChartEvent(
   const int     id,       // identifiant d'évènement
   const long&   lparam,   // paramètre de type long
   const double  dparam,   // paramètre de type double
   const string  sparam    // paramètre de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long.

dparam

[in] Paramètre d'évènement de type double.

sparam

[in] Paramètre d'évènement de type string.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnBookEvent

Gestionnaire de l'évènement OnBookEvent.

```
virtual void  OnBookEvent(
   const string&     symbol       // symbole
)
```

Paramètres

symbol

[in] Symbole de l'évènementOnBookEvent.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

InitParameters

Initialise les paramètres de l'Expert Advisor.

```
virtual bool  InitParameters()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La fonction InitParameters() de la classe de base CExpert n'effectue aucun traitement et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

DeinitTrade

Dé-initialise l'objet Trade.

```
virtual void  DeinitTrade()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

DeinitSignal

Dé-initialise l'objet Signal.

```
virtual void  DeinitSignal()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

DeinitTrailing

Dé-initialise l'objet Suiveur.

```
virtual void  DeinitTrailing()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

DeinitMoney

Dé-initialise l'objet Money Management.

```
virtual void  DeinitMoney()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

DeinitIndicators

Dé-initialise tous les indicateurs et les séries de données.

```
virtual void  DeinitIndicators()
```

Valeur de retour

Aucune.

Note

Cette fonction dé-initialise également tous les indicateurs et les séries de données de tous les objets auxiliaires.

© 2000-2024, MetaQuotes Ltd.

Refresh

Met à jour toutes les données

```
virtual bool  Refresh()
```

Valeur de retour

vrai si le traitement d'un autre tick est recquis, faux sinon.

Note

Permet de déterminer le besoin de traitement d'un tick. Si besoin, cette fonction met à jour toutes les cotations, les séries de données et les données des indicateurs puis retourne vrai.

Implémentation

```
//+------------------------------------------------------------------+
//| Rafraîchissement des données à traiter                           |
//| ENTREE : aucune.                                                 |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::Refresh()
  {
   MqlDateTime time;
//--- rafraîchit les cotations
   if(!m_symbol.RefreshRates()) return(false);
//--- vérifie le besoin de traitement
   TimeToStruct(m_symbol.Time(),time);
   if(m_period_flags!=WRONG_VALUE && m_period_flags!=0)
      if((m_period_flags & TimeframesFlags(time))==0) return(false);
   m_last_tick_time=time;
//--- rafraîchit les indicateurs
   m_indicators.Refresh();
//--- ok
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

Processing

Algorithme principal de traitement.

```
virtual bool  Processing()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Effecute les étapes suivantes :

1. Vérifie l'existence d'une position ouverte sur le symbole. Si aucune position n'est trouvée, passe à l'étape №5.
2. Vérifie les conditions pour retourner une position ouverte (méthode CheckReverse()). Si la position a été retournée, la fonction se termine.
3. Vérifie les conditions pour fermer la position (méthode CheckClose()). Si la position a été clôturée, passe à l'étape №5.
4. Vérifie les conditions pour modifier les paramètres de la position (méthode CheckTrailingStop()). Si les paramètres de la position ont été modifiés, la fonction se termine.
5. Vérifie l'existence d'ordres en attente sur le symbole. Si aucun ordre en attente n'est trouvé, passe à l'étape №9.
6. Vérifie les conditions pour supprimer un ordre (méthode CheckDeleteOrderLong() pour les ordres en attente de type buy ou CheckDeleteOrderShort() pour les ordres en attente de type sell). Si l'ordre a été supprimé, passe à l'étape №9.
7. Vérifie les conditions pour modifier les paramètres d'un ordre en attente (CheckTrailingOrderLong() pour les ordres de type buy ou CheckTrailingOrderShort() pour les ordres de type sell). Si les paramètres de l'ordre ont été modifiés, la fonction se termine.
8. Sortie.
9. Vérifie les conditions pour ouvrir une position (méthode CheckOpen()).

Si vous voulez implémenter votre propre algorithem, vous devez surcharger la méthode Processing() de la classe dérivée.

Implémentation

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Fonction principale                                              |
//| ENTREE : aucune.                                                 |
//| SORTIE : vrai si une opération a été effectuée, faux sinon.      |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::Processing()
  {
//--- vérifie l'existence de positions ouvertes
   if(m_position.Select(m_symbol.Name()))
     {
      //--- ouvre la position
      //--- vérifie la possibilité de renverser la position
      if(CheckReverse()) return(true);
      //--- vérifie la possibilité de fermer la position/supprimer les ordres en atter
      if(!CheckClose())
        {
         //--- vérifie la possibilité de modifier la position
         if(CheckTrailingStop()) return(true);
         //--- retour sans traitement
         return(false);
        }
     }
//--- vérifie la présence d'ordres en attente
   int total=OrdersTotal();
   if(total!=0)
     {
      for(int i=total-1;i>=0;i--)
        {
         m_order.SelectByIndex(i);
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         if(m_order.OrderType()==ORDER_TYPE_BUY_LIMIT || m_order.OrderType()==ORDER_TY
           {
            //--- vérifie la possibilité de supprimer un ordre long en attente
            if(CheckDeleteOrderLong()) return(true);
            //--- vérifie la possibilité de modifier un ordre long en attente
            if(CheckTrailingOrderLong()) return(true);
           }
         else
           {
            //--- vérifie la possibilité de supprimer un ordre sell en attente
            if(CheckDeleteOrderShort()) return(true);
            //--- vérifie la possibilité de modifier un ordre sell en attente
            if(CheckTrailingOrderShort()) return(true);
           }
         //--- retour sans traitement
         return(false);
        }
     }
//--- vérifie la possibilité d'ouvrir une position / de mettre un ordre en attente
   if(CheckOpen()) return(true);
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

SelectPosition

Selects a position to work with.

```
void  SelectPosition()
```

Return Value

No.

Implementation

```
//+------------------------------------------------------------------+
//| Position select                                                  |
//| INPUT:  no.                                                      |
//| OUTPUT: no.                                                      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::SelectPosition(void)
  {
   bool res=false;
//---
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      res=m_position.SelectByMagic(m_symbol.Name(),m_magic);
   else
      res=m_position.Select(m_symbol.Name());
//---
   return(res);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckOpen

Vérifie les conditions pour ouvrir une position.

```
virtual bool  CheckOpen()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

Vérifie les conditions pour ouvrir des positions longues (CheckOpenLong()) et courtes ou short (CheckOpenShort()).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour ouvrir une position ou définir un ordre limit/stop |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                              |
//+------------------------------------------------------------------+
bool CExpert::CheckOpen()
  {
   if(CheckOpenLong())  return(true);
   if(CheckOpenShort()) return(true);
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Vérifie les conditions pour ouvrir une position longue.

```
virtual bool  CheckOpenLong()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

La fonction vérifie les conditions pour ouvrir une position longue (méthode CheckOpenLong() de l'objet Signal) et ouvre la position longue (méthode OpenLong()) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour ouvrir une position longue / définir un ordre limit/stop |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- vérifie le signal pour ouvrir une opération longue
   if(m_signal.CheckOpenLong(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenLong(price,sl,tp));
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Vérifie les conditions pour ouvrir une position courte.

```
virtual bool  CheckOpenShort()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

La fonction vérifie les conditions pour ouvrir une position courte ou short (méthode CheckOpenShort() de l'objet Signal) et ouvre cette position (méthode OpenShort()) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour ouvrir une position short ou définir / ordre limit/stop |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- vérifie le signal pour ouvrir une opération short
   if(m_signal.CheckOpenShort(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenShort(price,sl,tp));
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

OpenLong

Ouvre une position longue.

```
virtual bool  OpenLong(
   double    price,     // prix
   double    sl,        // Stop Loss
   double    tp         // Take Profit
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

tp

[in] Prix du Take Profit

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Retourne le volume de trading (méthode LotOpenLong(...)) et ouvre une position long (méthode Buy() de l'objet Trade) si le volume de trading n'est pas égal à 0.

Implémentation

```
//+------------------------------------------------------------------+
//| Ouvre une position long ou définit un ordre limit/stop.          |
//| ENTREE :  price - prix,                                          |
//|         sl    - stop loss,                                      |
//|         tp    - take profit.                                    |
//| SORTIE : vrai si opération réalisée avec succès, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::OpenLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- récupère le lot pour l'ouverture
   double lot=LotOpenLong(price,sl);
//--- vérifie le lot pour l'ouverture
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

OpenShort

Ouvre une position courte (short)

```
virtual bool  OpenShort(
   double    price,     // prix
   double    sl,        // Stop Loss
   double    tp         // Take Profit
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

tp

[in] Prix du Take Profit

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Retourne le volume de trading (méthode LotOpenShort(...)) et ouvre une position short (méthode Sell() de l'objet Trade) si le volume de trading n'est pas égal à 0.

Implémentation

```
//+------------------------------------------------------------------+
//| Ouvre une position short ou définit un ordre limit/stop.    |
//| ENTREE :  price - prix,                                          |
//|         sl    - stop loss,                                      |
//|         tp    - take profit.                                    |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon.     |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::OpenShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- retourne le lot pour l'ouverture
   double lot=LotOpenShort(price,sl);
//--- vérifie le lot pour l'ouverture
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckReverse

Vérifie les conditions pour renverser une position ouverte.

```
virtual bool  CheckReverse()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

Vérifie les conditions pour renverser une position longue (CheckReverseLong()) ou courte (CheckReverseShort()).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérification pour renverser une position                         |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckReverse()
  {
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- vérifie la possibilité de renverser une position longue
      if(CheckReverseLong())  return(true);
     }
   else
      //--- vérifie la possibilité de renverser une position courte
      if(CheckReverseShort()) return(true);
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckReverseLong

Vérifie les conditions pour renverser une position longue.

```
virtual bool  CheckReverseLong()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

La fonction vérifie les conditions pour renverser une position longue (méthode CheckReverseLong() de l'objet Signal) et effectue le renversement de la position longue actuelle (méthode ReverseLong(...)) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Vérification pour renverser une position longue                  |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- vérifie le signal de retournement d'une position longue
   if(m_signal.CheckReverseLong(price,sl,tp,expiration)) return(ReverseLong(price,sl,t
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckReverseShort

Vérifie les conditions pour renverser une position courte (short).

```
virtual bool  CheckReverseLong()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

La fonction vérifie les conditions pour renverser une position courte (ou short) (méthode CheckReverseShort() de l'objet Signal) et effectue le renversement de la position short actuelle (méthode ReverseShort(...)) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Vérification pour renverser une position courte                  |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- vérifie le signal de retournement d'une position short
   if(m_signal.CheckReverseShort(price,sl,tp,expiration)) return(ReverseShort(price,sl
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

ReverseLong

Effectue le renversement d'une position longue.

```
virtual bool  ReverseLong(
   double    price,     // prix
   double    sl,        // Stop Loss
   double    tp         // Take Profit
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

tp

[in] Prix du Take Profit

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Récupère le volume du renversement de la position (méthode LotReverse()) et effectue le retournement de la position longue (méthode Sell() de l'objet Trade) si le volume de trading n'est pas égal à 0.

Implémentation

```
//+------------------------------------------------------------------+
//| Renversement d'une position longue                               |
//| ENTREE :  price - prix,                                         |
//|          sl    - stop loss,                                     |
//|          tp    - take profit.                                   |
//| SORTIE : vrai si opération réalisée avec succès, sinon faux. |
//| REMARQUE : aucune.                                              |
//+------------------------------------------------------------------+
bool CExpert::ReverseLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- récupère le lot pour le renversement
   double lot=LotReverse(sl);
//--- vérifie le lot
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

ReverseShort

Effectue le renversement d'une position courte (short).

```
virtual bool  ReverseShort(
   double    price,     // prix
   double    sl,        // Stop Loss
   double    tp         // Take Profit
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

tp

[in] Prix du Take Profit

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Récupère le volume du renversement de la position (méthode LotReverse()) et effectue le retournement de la position courte (méthode Buy() de l'objet Trade) si le volume de trading n'est pas égal à 0.

Implémentation

```
//+------------------------------------------------------------------+
//| Renversement d'une position courte                               |
//| ENTREE :  price - prix,                                          |
//|           sl    - stop loss,                                     |
//|           tp    - take profit.                                   |
//| SORTIE : vrai si opération réalisée avec succès, sinon faux. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::ReverseShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- récupère le lot pour le renversement
   double lot=LotReverse(sl);
//--- vérifie le lot
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckClose

Vérifie les conditions de clôture d'une position.

```
virtual bool  CheckClose()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

1. Cette fonction vérifie les conditions de sortie (Stop Out) de l'Expert Advisor (méthode CheckClose() de l'objet money management). Si la condition est satisfaite, cette fonction ferme la position et supprime tous les ordres (CloseAll(...)) et sort.
2. Elle vérifie les conditions pour fermer une position longue ou courte (méthodes CheckCloseLong() ou CheckCloseShort()) et si la position est fermée, elle supprime tous les ordres (méthode DeleteOrders()).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour fermer une position ou supprimer un ordre limite/stop|
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckClose()
  {
   double lot;
//--- la position doit être sélectionnée avant l'appel
   if((lot=m_money.CheckClose(GetPointer(m_position)))!=0.0)
      return(CloseAll(lot));
//--- vérifie le type de position
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- vérifie la possibilité de fermer la position longue / supprimer les ordres
      if(CheckCloseLong())
        {
         DeleteOrders();
         return(true);
        }
     }
   else
     {
      //--- vérifie la possibilité de fermer la position courte / supprimer les ordres
      if(CheckCloseShort())
        {
         DeleteOrders();
         return(true);
        }
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckCloseLong

Vérifie les conditions pour clôturer une position longue.

```
virtual bool  CheckCloseLong()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction vérifie les conditions pour clôturer une position longue (méthode CheckCloseLong() de l'objet Signal). Si les conditions sont réunies, elle ferme la position ouverte (méthode CloseLong(...)).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour fermer une position longue ou supprimer un ordre limit/stop |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseLong()
  {
   double price=EMPTY_VALUE;
//--- vérifie les opérations pour clôturer les opérations sur les positions longues
   if(m_signal.CheckCloseLong(price))
      return(CloseLong(price));
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckCloseShort

Vérifie les conditions pour clôturer une position courte (short).

```
virtual bool  CheckCloseShort()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction vérifie les conditions pour clôturer une position courte (ou short) (méthode CheckCloseShort() de l'objet Signal). Si les conditions sont réunies, elle ferme la position ouverte (méthode CloseShort(...)).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour fermer une position courte ou supprimer un ordre limit/stop |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseShort()
  {
   double price=EMPTY_VALUE;
//--- vérifie les opérations pour clôturer les opérations sur les positions short
   if(m_signal.CheckCloseShort(price))
      return(CloseShort(price));
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CloseAll

Ferme une partie ou toutes les positions.

```
virtual bool  CloseAll(
   double    lot    // lot
   )
```

Paramètres

lot

[in] Nombre de lots pour réduire la position.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Dans le mode "netting", une position est fermée avec les méthodes CExpertTrade::Buy ou CExpertTrade::Sell. Dans le mode "hedging", la méthode CTrade::PositionClose est utilisée, qui peut également être utilisée sur les comptes ayant le mode netting. La méthode DeleteOrders() est utilisée pour supprimer des ordres.

Implémentation

```
//+------------------------------------------------------------------+
//| Clôture d'une position et suppression des ordres                 |
//| ENTREE :  lot - volume à clôturer.                               |
//| SORTIE : vrai si l'opération a été réalisée avec succès, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CloseAll(double lot)
  {
   bool result;
//--- vérifie les opérations
   if(m_position.PositionType()==POSITION_TYPE_BUY) result=m_trade.Sell(lot,0,0,0);
   else                                             result=m_trade.Buy(lot,0,0,0);
   result|=DeleteOrders();
//---
   return(result);
  }
```

© 2000-2024, MetaQuotes Ltd.

Close

Ferme les positions ouvertes.

```
virtual bool  Close()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Ferme la position (méthode PositionClose() de l'objet CTrade).

Implémentation

```
//+-----------------------------------------------------------------+
//| Ferme la position |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                              |
//+-----------------------------------------------------------------+
bool CExpert::Close()
  {
   return(m_trade.PositionClose(m_symbol.Name()));
  }
```

© 2000-2024, MetaQuotes Ltd.

CloseLong

Ferme la position longue.

```
virtual bool  CloseLong(
   double    price    // prix
   )
```

Paramètres

price

[in] Prix.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Ferme la position long (méthode Sell(...) de l'objet CTrade).

Implémentation

```
//+-----------------------------------------------------------------+
//| Ferme une position longue                                       |
//| ENTREE :  price - prix de clôture. |
//| SORTIE : vrai si l'opération a été réalisée avec succès, sinon faux. |
//| REMARQUE : aucune. |
//+-----------------------------------------------------------------+
bool CExpert::CloseLong(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Sell(m_position.Volume(),price,0,0));
  }
```

© 2000-2024, MetaQuotes Ltd.

CloseShort

Ferme la position courte (short)

```
virtual bool  CloseShort(
   double    price    // prix
   )
```

Paramètres

price

[in] Prix.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Ferme la position courte (short) (méthode Buy(...) de l'objet CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Ferme une position short                                         |
//| ENTREE :  price - prix de clôture. |
//| SORTIE : vrai si opération réalisée avec succès, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CloseShort(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Buy(m_position.Volume(),price,0,0));
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStop

Vérifie les conditions du Stop Suiveur (Trailing Stop) de la position ouverte.

```
virtual bool  CheckTrailingStop()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

Vérifie les conditions du Stop Suiveur (Trailing Stop) de la position ouverte (CheckTrailingStopLong() ou CheckTrailingStopShort() pour les positions longues ou courtes).

Implémentation

```
//+------------------------------------------------------------------+
//| Check for trailing stop/profit position                          |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStop()
  {
//--- la position doit d'abord être sélectionnée
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- vérifie la possibilité de modifier une position long
      if(CheckTrailingStopLong()) return(true);
     }
   else
     {
      //--- vérifie la possibilité de modifier une position short
      if(CheckTrailingStopShort()) return(true);
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Cette fonction vérifie les conditions du Stop Suiveur (Trailing Stop) de la position long ouverte.

```
virtual bool  CheckTrailingStopLong()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Cette fonction vérifie les conditions du Stop Suiveur (Trailing Stop) de la position long ouverte (méthode CheckTrailingStopLong(...) de l'objet Expert Trailing). Si les conditions sont réunies, elle modifie les paramètres de la position (méthode TrailingStopLong(...)).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérifie le stop/profit suiveur d'une position longue |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopLong()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- vérifie les opérations des stops suiveur des opérations long
   if(m_trailing.CheckTrailingStopLong(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- traitement du stop suiveur de l'opération
      return(TrailingStopLong(sl,tp));
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Cette fonction vérifie les conditions du Stop Suiveur (Trailing Stop) de la position courte ouverte.

```
virtual bool  CheckTrailingStopShort()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Cette fonction vérifie les conditions du Stop Suiveur (Trailing Stop) de la position courte ouverte (méthode CheckTrailingStopShort(...) de l'objet Expert Trailing). Si les conditions sont réunies, elle modifie les paramètres de la position (méthode TrailingStopShort(...)).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérifie le stop/profit suiveur d'une position short |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopShort()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- vérifie les opérations des stops suiveur des opérations short
   if(m_trailing.CheckTrailingStopShort(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- traitement du stop suiveur de l'opération short
      return(TrailingStopShort(sl,tp));
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

TrailingStopLong

Modifie les paramètres d'une position longue ouverte.

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Paramètres

sl

[in] Prix du Stop Loss

tp

[in] Prix du Take Profit

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction modifie les paramètres de la position longue ouverte (méthode PositionModify(...) de la classe d'objet CTrade).

Implémentation

```
//+-----------------------------------------------------------------+
//| Stop/profit suiveur pour une position longue                    |
//| ENTREE :  sl - nouveau stop loss,                               |
//|         tp - nouveau take profit.                               |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune.                          |
//+-----------------------------------------------------------------+
bool CExpert::TrailingStopLong(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

TrailingStopShort

Modifie les paramètres d'une position courte ouverte.

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Paramètres

sl

[in] Prix du Stop Loss

tp

[in] Take Profit price.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction modifie les paramètres de la position courte ouverte (méthode PositionModify(...) de la classe d'objet CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Stop/profit suiveur pour une position short                      |
//| ENTREE :  sl - nouveau stop loss,                                |
//|         tp - nouveau take profit.                                |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopShort(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckTrailingOrderLong

Vérifie les conditions du Stop Suiveur (Trailing Stop) d'un ordre en attente de type Buy Limit/Stop

```
virtual bool  CheckTrailingOrderLong()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction vérifie les conditions du Stop Suiveur (Trailing Stop) pour un ordre Buy Limit/Stop (méthode CheckTrailingOrderLong() de l'objet Signal de Trading). Si les conditions sont réunies, les paramètres de l'ordre sont modifiés (méthode TrailingOrderLong(...)) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Check for trailing long limit/stop order                         |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderLong()
  {
   double price;
//--- vérifie la possibilité de modifier un ordre long
   if(m_signal.CheckTrailingOrderLong(GetPointer(m_order),price))
      return(TrailingOrderLong(m_order.PriceOpen()-price));
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckTrailingOrderShort

Vérifie les conditions du Stop Suiveur (Trailing Stop) d'un ordre en attente de type Sell Limit/Stop

```
virtual bool  CheckTrailingOrderShort()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction vérifie les conditions du Stop Suiveur (Trailing Stop) pour un ordre Sell Limit/Stop (méthode CheckTrailingOrderShort() de l'objet Signal de Trading). Si les conditions sont réunies, les paramètres de l'ordre sont modifiés (méthode TrailingOrderShort(...)) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Check for trailing short limit/stop order                        |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderShort()
  {
   double price;
//--- vérifie la possibilité de modifier un ordre short
   if(m_signal.CheckTrailingOrderShort(GetPointer(m_order),price))
      return(TrailingOrderShort(m_order.PriceOpen()-price));
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

TrailingOrderLong

Modifie les paramètres d'un ordre en attente Buy Limit/Stop.

```
virtual bool  TrailingOrderLong(
   double    delta    // delta
   )
```

Paramètres

delta

[in] Prix delta.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Modifie les paramètres d'un ordre en attente Buy Limit/Stop (méthode OrderModify(...) de la classe CTrade).

Implémentation

```
//+-----------------------------------------------------------------+
//| Ordre long limit/stop en attente |
//| ENTREE :  delta - changement de prix.                           |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune.                          |
//+-----------------------------------------------------------------+
bool CExpert::TrailingOrderLong(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modification de l'ordre long
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

© 2000-2024, MetaQuotes Ltd.

TrailingOrderShort

Modifie les paramètres d'un ordre en attente Sell Limit/Stop.

```
virtual bool  TrailingOrderShort(
   double    delta    // delta
   )
```

Paramètres

delta

[in] Prix delta.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Modifie les paramètres d'un ordre en attente Sell Limit/Stop (méthode OrderModify(...) de la classe CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Ordre short limit/stop en attente                                |
//| ENTREE :  delta - changement de prix.                            |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderShort(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modification de l'ordre short
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckDeleteOrderLong

Vérifie les conditions pour supprimer un ordre en attente de type Buy Limit/Stop.

```
virtual bool  CheckDeleteOrderLong()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Cette fonction vérifie l'heure d'expiration d'un ordre. La fonction vérifie les conditions pour supprimer un ordre Sell Limit/Stop (méthode CheckCloseLong() de l'objet Signal). Si les conditions sont réunies, elle supprime l'ordre (méthode DeleteOrderLong()).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour supprimer un ordre long limit/stop                   |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderLong()
  {
   double price;
//--- vérifie la possibilité de supprimer un ordre long
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderLong());
     }
   if(m_signal.CheckCloseLong(price))
      return(DeleteOrderLong());
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckDeleteOrderShort

Vérifie les conditions pour supprimer un ordre en attente de type Sell Limit/Stop.

```
virtual bool  CheckDeleteOrderShort()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Cette fonction vérifie l'heure d'expiration d'un ordre. La fonction vérifie les conditions pour supprimer un ordre en attente Sell Limit/Stop (méthode CheckCloseShort() de l'objet Signal). Si les conditions sont réunies, l'ordre est supprimé (méthode DeleteOrderShort()).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour supprimer un ordre short limit/stop |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                              |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderShort()
  {
   double price;
//--- vérifie la possibilité de supprimer un ordre short
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderShort());
     }
   if(m_signal.CheckCloseShort(price))
      return(DeleteOrderShort());
//--- retour sans traitement
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteOrders

Supprime tous les ordres.

```
virtual bool  DeleteOrders()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Supprime tous les ordres (DeleteOrder() pour tous les ordres).

Implémentation

```
//+------------------------------------------------------------------+
//| Supprime tous les ordres limite/stop                             |
//| ENTREE : aucune.                                                 |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrders()
  {
   bool result=false;
   int  total=OrdersTotal();
//---
   for(int i=total-1;i>=0;i--)
     {
      if(m_order.Select(OrderGetTicket(i)))
        {
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         result|=DeleteOrder();
        }
     }
//---
   return(result);
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteOrder

Supprime un ordre en attete Limit/Stop.

```
virtual bool  DeleteOrder()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Supprime l'ordre en attente Limit/Stop (méthode OrderDelete(...) de la classe CTrade).

Implémentation

```
//+-----------------------------------------------------------------+
//| Supprime un ordre limit/stop                                    |
//| ENTREE : aucune.                                                |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune.                                              |
//+-----------------------------------------------------------------+
bool CExpert::DeleteOrder()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteOrderLong

Supprime un ordre en attente de type Buy Limit/Stop.

```
virtual bool  DeleteOrderLong()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Supprime l'ordre en attente Buy Limit/Stop (méthode OrderDelete(...) de la classe CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Supprime un ordre long limit/stop                                |
//| ENTREE : aucune.                                                 |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderLong()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteOrderShort

Supprime un ordre en attente de type Sell Limit/Stop.

```
virtual bool  DeleteOrderShort()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Supprime l'ordre en attente Sell Limit/Stop (méthode OrderDelete(...) de la classe CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Supprime l'ordre short limit/stop                                |
//| ENTREE : aucune.                                                  |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderShort()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2024, MetaQuotes Ltd.

LotOpenLong

Retourne le volume de trading pour une opération de type buy

```
double  LotOpenLong(
   double    price,    // prix
   double    sl        // Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading (en lots) pour une opération de type buy.

Note

Cette fonction retourne le volume de trading pour une opération de type buy (méthode CheckOpenLong(...) de l'objet money management).

Implémentation

```
//+-----------------------------------------------------------------+
//| Méthode pour récupérer le nb de lots d'une position long ouverte. |
//| ENTREE :  price - prix,                                          |
//|         sl    - stop loss.                                       |
//| SORTIE : nb de lots.                                             |
//| REMARQUE : aucune. |
//+-----------------------------------------------------------------+
double CExpert::LotOpenLong(double price,double sl)
  {
   return(m_money.CheckOpenLong(price,sl));
  }
```

© 2000-2024, MetaQuotes Ltd.

LotOpenShort

Retourne le volume de trading pour une opération de type sell.

```
double  LotOpenShort(
   double    price,    // prix
   double    sl        // Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss.

Valeur de retour

Volume de trading (en lots) pour une opération de type sell.

Note

Cette fonction retourne le volume de trading pour une opération de type sell (méthode CheckOpenShort(...) de l'objet money management).

Implémentation

```
//+-----------------------------------------------------------------+
//| Méthode pour récupérer le nb de lots d'une position short ouverte.              |
//| ENTREE :  price - prix,                                          |
//|         sl    - stop loss.                                      |
//| SORTIE : nb de lots.                                             |
//| REMARQUE : aucune. |
//+-----------------------------------------------------------------+
double CExpert::LotOpenShort(double price,double sl)
  {
   return(m_money.CheckOpenShort(price,sl));
  }
```

© 2000-2024, MetaQuotes Ltd.

LotReverse

Retourne le volume de trading pour une opération de renversement d'une position.

```
double  LotReverse(
   double    sl       // Stop Loss
   )
```

Paramètres

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading (en lots) pour une opération de renversement de position.

Note

Cette fonction retourne le volume de trading pour une opération de renversement de position (méthode CheckReverse(...) de l'objet money management).

Implémentation

```
//+-----------------------------------------------------------------+
//| Méthode pour récupérer le nb de lots pour un renversement de position. |
//| ENTREE :  sl - stop loss.                                        |
//| SORTIE : nb de lots.                                            |
//| REMARQUE : aucune. |
//+-----------------------------------------------------------------+
double CExpert::LotReverse(double sl)
  {
   return(m_money.CheckReverse(GetPointer(m_position),sl));
  }
```

© 2000-2024, MetaQuotes Ltd.

PrepareHistoryDate

Définit la date de début de suivi de l'historique de trading.

```
void  PrepareHistoryDate()
```

Note

La période de suivi de l'historique de trading est défini à partir du début du mois (mais pas moins de 1 jour).

© 2000-2024, MetaQuotes Ltd.

HistoryPoint

Crée un point de contrôle de l'historique de trading (sauvegarde le nombre de positions, les ordres, les deals et les ordres historiques).

```
void  HistoryPoint(
   bool   from_check_trade=false   // flag
   )
```

Paramètres

from_check_trade=false

[in] Flag pour éviter la récursion.

Note

Sauvegarde le montant des positions, des ordres, des deals et l'historique des transactions.

© 2000-2024, MetaQuotes Ltd.

CheckTradeState

Compare l'état courant avec celui sauvegardé et appelle le gestionnaire d'évènement correspondant.

```
bool  CheckTradeState()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

Elle vérifie le nombre de positions, d'ordres, de deals et les ordres historiques en comparant les valeurs sauvegardées par la méthode HistoryPoint() . Si l'historique de trading a changé, cette fonction appelle le gestionnaire d'évènement virtuel correspondant.

© 2000-2024, MetaQuotes Ltd.

WaitEvent

Définit le flag d'attente d'évènement.

```
void  WaitEvent(
   ENUM_TRADE_EVENTS    event         // flag
   )
```

Paramètres

event

[in] Flag avec la liste d'évènements à définir (énumération ENUM_TRADE_EVENTS).

Valeur de retour

Aucune.

Flags d'Evènement

```
//--- flags des évènements attendus
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT              =0,         // aucun évènement attendu
   TRADE_EVENT_POSITION_OPEN         =0x1,       // en attente de l'évènement "ouvertu
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,       // en attente de l'évènement "modific
   TRADE_EVENT_POSITION_MODIFY       =0x4,       // en attente de l'évènement "modific
   TRADE_EVENT_POSITION_CLOSE        =0x8,       // en attente de l'évènement "clôture
   TRADE_EVENT_POSITION_STOP_TAKE    =0x10,      // en attente de l'évènement "déclenc
   TRADE_EVENT_ORDER_PLACE           =0x20,      // en attente de l'évènement "placeme
   TRADE_EVENT_ORDER_MODIFY          =0x40,      // en attente de l'évènement "modific
   TRADE_EVENT_ORDER_DELETE          =0x80,      // en attente de l'évènement "suppres
   TRADE_EVENT_ORDER_TRIGGER         =0x100      // en attente de l'évènement "déclenc
  };
```

© 2000-2024, MetaQuotes Ltd.

NoWaitEvent

Réinitialise le flag d'attente d'évènement.

```
void  NoWaitEvent(
   ENUM_TRADE_EVENTS    event         // flag
   )
```

Paramètres

event

[in] Flag avec la liste d'évènements à réinitialiser (énumération ENUM_TRADE_EVENTS).

Valeur de retour

Aucune.

Flags d'Evènement

```
//--- flags des évènements attendus
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT              =0,         // aucun évènement attendu
   TRADE_EVENT_POSITION_OPEN         =0x1,       // en attente de l'évènement "ouvertu
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,       // en attente de l'évènement "modific
   TRADE_EVENT_POSITION_MODIFY       =0x4,       // en attente de l'évènement "modific
   TRADE_EVENT_POSITION_CLOSE        =0x8,       // en attente de l'évènement "clôture
   TRADE_EVENT_POSITION_STOP_TAKE    =0x10,      // en attente de l'évènement "déclenc
   TRADE_EVENT_ORDER_PLACE           =0x20,      // en attente de l'évènement "placeme
   TRADE_EVENT_ORDER_MODIFY          =0x40,      // en attente de l'évènement "modific
   TRADE_EVENT_ORDER_DELETE          =0x80,      // en attente de l'évènement "suppres
   TRADE_EVENT_ORDER_TRIGGER         =0x100      // en attente de l'évènement "déclenc
  };
```

© 2000-2024, MetaQuotes Ltd.

TradeEventPositionStopTake

Gestionnaire de l'évènement "Stop Loss/Take Profit d'une position déclenché".

```
virtual bool  TradeEventPositionStopTake()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

TradeEventOrderTriggered

Gestionnaire de l'évènement "Ordre en attente déclenché".

```
virtual bool  TradeEventOrderTriggered()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

TradeEventPositionOpened

Gestionnaire de l'évènement "Position ouverte".

```
virtual bool  TradeEventPositionOpened()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

TradeEventPositionVolumeChanged

Gestionnaire de l'évènement "Volume de position changé".

```
virtual bool  TradeEventPositionVolumeChanged()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

TradeEventPositionModified

Gestionnaire de l'évènement "Position modifiée".

```
virtual bool  TradeEventPositionModified()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

TradeEventPositionClosed

Gestionnaire de l'évènement "Position fermée".

```
virtual bool  TradeEventPositionClosed()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

TradeEventOrderPlaced

Gestionnaire de l'évènement "Ordre en attente placé".

```
virtual bool  TradeEventOrderPlaced()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

TradeEventOrderModified

Gestionnaire de l'évènement "Ordre en attente modifié".

```
virtual bool  TradeEventOrderModified()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

TradeEventOrderDeleted

Gestionnaire de l'évènement "Ordre enattente supprimé".

```
virtual bool  TradeEventOrderDeleted()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

TradeEventNotIdentified

Gestionnaire d'un évènement non identifié.

```
virtual bool  TradeEventNotIdentified()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

Note

Différents évènements de trading peuvent arriver, dans ce cas, il est difficile de les identifier.

© 2000-2024, MetaQuotes Ltd.

TimeframeAdd

Ajoute une période à surveiller.

```
void  TimeframeAdd(
   ENUM_TIMEFRAMES    period         // période
   )
```

Paramètres

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

TimeframesFlags

La méthode retourne le flag indiquant les périodes d'une nouvelle barre.

```
int  TimeframesFlags(
   MqlDateTime&    time         // variable de date/heure
   )
```

Paramètres

time

[in] Variable de type MqlDateTime pour la nouvelle date/heure, passée par référence.

Valeur de retour

Retourne le flag indiquant les périodes d'une nouvelle barre.

© 2000-2024, MetaQuotes Ltd.

CExpertSignal

CExpertSignal est la classe de base pour les signaux de trading, elle ne fait rien (exceptées les méthodes CheckReverseLong() et CheckReverseShort()) mais fournit l'interface.

Comment l'utiliser :

1. Préparez un algorithme pour les signaux de trading ;
2. Créez votre propre classe de signal de trading, dérivant de la classe CExpertSignal ;
3. Surchargez les méthodes virtuelles de votre classe avec vos propres algorithmes.

Vous pouvez trouver des exemples des classes de signaux de trading dans le répertoire Expert\Signal\.

Description

CExpertSignal est la classe de base pour l'implémentation des algorithmes des signaux de trading.

Déclaration

```
class CExpertSignal : public CExpertBase
```

Titre

```
#include <Expert\ExpertSignal.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertSignal

Descendants directs

CSignalAC, CSignalAMA, CSignalAO, CSignalBearsPower, CSignalBullsPower, CSignalCCI, CSignalDeM, CSignalDEMA, CSignalEnvelopes, CSignalFrAMA, CSignalRSI, CSignalRVI, CSignalSAR, CSignalStoch, CSignalTEMA, CSignalTriX, CSignalWPR

Méthodes de Classe

Initialisation	
virtual InitIndicators	Initialise les indicateurs et les séries de données.
virtual ValidationSettings	Vérifie les paramètres
virtual AddFilter	Ajoute un filtre au signal combiné
Accés aux Données Protégées	
BasePrice	Définit le niveau du prix de base
UsedSeries	Retourne les flags des séries de données utilisées
Définition des Paramètres	

© 2000-2024, MetaQuotes Ltd.

Initialisation	
Weight	Définit la valeur du paramètres "Weight" (poids)
PatternsUsage	Définit la valeur du paramètre "PatternsUsage"
General	Définit la valeur du paramètre "General"
Ignore	Définit la valeur du paramètre "Ignore"
Invert	Définit la valeur du paramètre "Invert"
ThresholdOpen	Définit la valeur du paramètre "ThresholdOpen"
ThresholdClose	Définit la valeur du paramètre "ThresholdClose"
PriceLevel	Définit la valeur du paramètre "PriceLevel"
StopLevel	Définit la valeur du paramètre "StopLevel"
TakeLevel	Définit la valeur du paramètre "TakeLevel"
Expiration	Définit la valeur du paramètre "Expiration"
Magic	Définit la valeur du paramètre "Magic"
Vérifier les Conditions de Trading	
virtual CheckOpenLong	Vérifie les conditions d'ouverture d'une position longue
virtual CheckCloseLong	Vérifie les conditions pour clôturer une position longue
virtual CheckOpenShort	Vérifie les conditions d'ouverture d'une position courte (short)
virtual CheckCloseShort	Vérifie les conditions pour clôturer une position courte (short)
virtual CheckReverseLong	Vérifie les conditions pour renverser une position longue
virtual CheckReverseShort	Vérifie les conditions pour renverser une position courte
Définir les Paramètres de Trading	
virtual OpenLongParams	Définit les paramètres pour ouvrir une position longue
virtual OpenShortParams	Définit les paramètres pour ouvrir une position courte
virtual CloseLongParams	Définit les paramètres pour fermer une position longue
virtual CloseShortParams	Définit les paramètres pour fermer une position courte
Vérifier les Conditions des Ordres Suiveurs	
virtual CheckTrailingOrderLong	Vérifie les conditions pour modifier les paramètres d'un ordre d'achat (Buy) en attente
virtual CheckTrailingOrderShort	Vérifie les conditions pour modifier les paramètres d'un ordre de vente (Sell) en attente

© 2000-2024, MetaQuotes Ltd.

Initialisation	
Méthodes pour Vérifier les Ordres de Marché	
virtual LongCondition	Retourne le résultat de la vérification des conditions d'achat
virtual ShortCondition	Retourne le résultat de la vérification des conditions de vente
virtual Direction	Retourne la direction "pondérée" du prix

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2024, MetaQuotes Ltd.

BasePrice

Définit le niveau du prix de base.

```
void  BasePrice(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du niveau du prix de base.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

UsedSeries

Retourne les flags des séries de données utilisées.

```
int  UsedSeries()
```

Valeur de retour

Les flags des séries de données utilisées (si le symbole et la période correspondent au symbole et à la période en cours), sinon 0.

© 2000-2024, MetaQuotes Ltd.

Weight

Définit la nouvelle valeur du paramètre "Weight".

```
void  Weight(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "Weight".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

PatternUsage

Définit la nouvelle valeur du paramètre "PatternsUsage".

```
void  PatternUsage(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "PatternsUsage".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

General

Définit la nouvelle valeur du paramètre "General".

```
void  General(
   int    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "General".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Ignore

Définit la nouvelle valeur du paramètre "Ignore".

```
void  Ignore(
   long    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "Ignore".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Invert

Définit la nouvelle valeur du paramètre "Invert".

```
void  Invert(
   long    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "Invert".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

ThresholdOpen

Définit la nouvelle valeur du paramètre "ThresholdOpen".

```
void  ThresholdOpen(
   long    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "ThresholdOpen".

Valeur de retour

Aucune.

Note

L'intervalle du paramètre "ThresholdOpen" est compris entre 0 et 100. Utilisé lors du "vote" pour ouvrir une position.

© 2000-2024, MetaQuotes Ltd.

ThresholdClose

Définit la valeur du paramètre "ThresholdClose".

```
void  ThresholdOpen(
   long    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "ThresholdClose".

Valeur de retour

Aucune.

Note

L'intervalle du paramètre "ThresholdClose" est compris entre 0 et 100. Utilisé lors du "vote" pour fermer une position.

© 2000-2024, MetaQuotes Ltd.

PriceLevel

Définit la nouvelle valeur du paramètre "PriceLevel"

```
void  PriceLevel(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "PriceLevel".

Valeur de retour

Aucune.

Note

La valeur du paramètre "PriceLevel" est définie en unités de prix. La valeur numérique de l'unité de prix est retournée par la méthode PriceLevelUnit() . Le "PriceLevel" est utilisé pour définir le prix d'ouverture relativement au prix de base.

© 2000-2024, MetaQuotes Ltd.

StopLevel

Définit la valeur du paramètre "StopLevel".

```
void  StopLevel(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "StopLevel".

Valeur de retour

Aucune.

Note

La valeur du paramètre "StopLevel" est définie en unités de prix. La valeur numérique de l'unité de prix est retournée par la méthode PriceLevelUnit() <t3>. Le "StopLevel" est utilisé pour définir le prix du Stop Loss relativement au prix d'ouverture.</t3>

© 2000-2024, MetaQuotes Ltd.

TakeLevel

Définit la valeur du paramètre "TakeLevel"

```
void  TakeLevel(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "TakeL".

Valeur de retour

Aucune.

Note

La valeur du paramètre "TakeLevel" est définie en unités de prix. La valeur numérique de l'unité de prix est retournée par la méthode PriceLevelUnit() <t3>. Le "TakeLevel" est utilisé pour définir le prix du Take Profit relativement au prix d'ouverture.</t3>

© 2000-2024, MetaQuotes Ltd.

Expiration

Définit la valeur du paramètre "Expiration".

```
void  Expiration(
   int    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "Expiration".

Valeur de retour

Aucune.

Note

La valeur du paramètre "Expiration" est définie en barres. Elle est utilisée comme date/heure d'expiration pour les ordres en attente (en cas de trading utilisant les ordres en attentes).

© 2000-2024, MetaQuotes Ltd.

Magic

Définit la valeur du paramètre "Magic".

```
void  Magic(
   int    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "Magic" (identifiant de l'Expert Advisor).

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

InitIndicators

Initialise tous les indicateurs et les séries de données.

```
virtual bool  InitIndicators(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Les séries de données ne sont initialisées que si l'objet utilise le symbole ou la période définit lors de l'initialisation et que le symbole ou la période sont différents.

© 2000-2024, MetaQuotes Ltd.

AddFilter

Ajoute un filtre au signal composite.

```
virtual bool  AddFilter(
   CExpertSignal*  filter    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur vers un objet de filtre.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Vérifie les conditions pour ouvrir une position longue.

```
virtual bool  CheckOpenLong(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Vérifie les conditions pour ouvrir une position courte.

```
virtual bool  CheckOpenShort(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OpenLongParams

Définit les paramètres pour ouvrir une position longue.

```
virtual bool  OpenLongParams(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OpenShortParams

Définit les paramètres pour ouvrir une position courte.

```
virtual bool  OpenShortParams(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CheckCloseLong

Vérifie les conditions pour clôturer une position longue.

```
virtual bool  CheckCloseLong(
   double&  price      // prix
   )
```

Paramètres

price

[in][out] Variable pour le prix de clôture, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CheckCloseShort

Vérifie les conditions pour clôturer une position courte (short).

```
virtual bool  CheckCloseShort(
   double&  price      // prix
   )
```

Paramètres

price

[in][out] Variable pour le prix de clôture, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CloseLongParams

Définit les paramètres pour fermer une position longue.

```
virtual bool  CloseLongParams(
   double&    price          // prix
   )
```

Paramètres

price

[in][out] Variable pour le prix de clôture, passée par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CloseShortParams

Définit les paramètres pour fermer une position courte.

```
virtual bool  CloseShortParams(
   double&    price          // prix
   )
```

Paramètres

price

[in][out] Variable pour le prix de clôture, passée par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CheckReverseLong

Vérifie les conditions pour renverser une position longue.

```
virtual bool  CheckReverseLong(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CheckReverseShort

Vérifie les conditions pour renverser une position courte.

```
virtual bool  CheckReverseShort(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le renversement de prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingOrderLong

Vérifie les conditions pour modifier les paramètres d'un ordre d'achat (Buy) en attente.

```
virtual bool  CheckTrailingOrderLong(
   COrderInfo*     order,          // ordre
   double&         price           // prix
   )
```

Paramètres

order

[in] Pointeur vers un objet de classe COrderInfo.

price

[in][out] Variable du prix du Stop Loss

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingOrderShort

Vérifie les conditions pour modifier les paramètres d'un ordre de vente (Sell) en attente.

```
virtual bool  CheckTrailingOrderShort(
   COrderInfo*     order,          // ordre
   double&         price           // prix
   )
```

Paramètres

order

[in] Pointeur vers un objet de classe COrderInfo.

price

[in][out] Variable du prix du Stop Loss

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2024, MetaQuotes Ltd.

LongCondition

Vérifie les conditions pour ouvrir une position longue.

```
virtual int  LongCondition()
```

Valeur de retour

Si les conditions sont réunies, retourne une valeur comprise entre 1 et 100 (dépendant de la "puissance" du signal), s'il n'y a aucun signal pour ouvrir une position longue, retourne 0.

Note

La méthode LongCondition() de la classe de base n'implémente pas la vérification des conditions pour ouvrir une position longue et retourne toujours 0.

© 2000-2024, MetaQuotes Ltd.

ShortCondition

Vérifie les conditions pour ouvrir une position courte.

```
virtual int  ShortCondition()
```

Valeur de retour

Si les conditions sont réunies, retourne une valeur comprise entre 1 et 100 (dépendant de la "puissance" du signal), s'il n'y a aucun signal pour ouvrir une position courte, retourne 0.

Note

La méthode ShortCondition() de la classe de base n'implémente pas la vérification des conditions pour ouvrir une position courte et retourne toujours 0.

© 2000-2024, MetaQuotes Ltd.

Direction

Retourne la valeur de la direction "pondérée".

```
virtual double  Direction()
```

Valeur de retour

Retourne une valeur supérieure à 0 si la direction est haussière (probablement) et inférieure à 0 si la direction est baissière. La valeur absolue dépend de la "puissance" du signal.

Note

Si les filtres sont utilisés, le résultat dépendra des filtres.

© 2000-2024, MetaQuotes Ltd.

CExpertTrailing

CExpertTrailing est la classe de base pour les algorithmes des objets suiveurs, elle est une interface et n'effectue rien.

Comment l'utiliser :

1. Préparez un algorithme pour le suivi ;
2. Creéez votre propre classe, dérivant de la classe CExpertTrailing ;
3. Surchargez les méthodes virtuelles de votre classe avec vos propres algorithmes.

Vous pouvez trouver des exemples de classes de suivi dans le répertoire Expert\Trailing\.

Description

CExpertTrailing est la classe de base pour l'implémentation d'algorithmes de stops suiveurs.

Déclaration

```
class CExpertTrailing : public CExpertBase
```

Titre

```
#include <Expert\ExpertTrailing.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertTrailing

Descendants directs

CTrailingFixedPips, CTrailingMA, CTrailingNone, CTrailingPSAR

Méthodes de Classe

Vérifier les Conditions du Stop Suiveur	
virtual CheckTrailingStopLong	Vérifie les conditions pour modifier les paramètres de la position longue
virtual CheckTrailingStopShort	Vérifie les conditions pour modifier les paramètres de la position courte

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Vérifie les conditions pour modifier les paramètres de la position longue.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // pointeur
   double&         sl,            // Stop Loss
   double&         tp             // Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet de classe CPositionInfo<t3>.</t3>

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

La méthode CheckTrailingStopLong() de la classe de base retourne toujours faux.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Vérifie les conditions pour modifier les paramètres de la position courte.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // pointeur
   double&         sl,            // Stop Loss
   double&         tp             // Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet de classe CPositionInfo<t3>.</t3>

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

La méthode CheckTrailingStopShort() de la classe de base retourne toujours faux.

© 2000-2024, MetaQuotes Ltd.

CExpertMoney

CExpertMoney est la classe de base pour les algorithmes de money et de risk management (gestion de votre capital et du risque).

Description

CExpertMoney est la classe de base pour implémenter les classes de money et de risk management.

Déclaration

```
class CExpertMoney : public CObject
```

Titre

```
#include <Expert\ExpertMoney.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertMoney

Descendants directs

CMoneyFixedLot, CMoneyFixedMargin, CMoneyFixedRisk, CMoneyNone, CMoneySizeOptimized

Méthodes de Classe

Accés aux Données Protégées	
Percent	Définit la valeur du paramètre "Pourcentage de risque"
Initialisation	
virtual ValidationSettings	Vérifie les paramètres
Vérification des Conditions de Trading	
virtual CheckOpenLong	Retourne le volume d'une position longue
virtual CheckOpenShort	Retourne le volume d'une position courte
virtual CheckReverse	Retourne le volume pour renverser une position
virtual CheckClose	Vérifie les conditions pour clôturer une position ouverte

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2024, MetaQuotes Ltd.

Percent

Définit la valeur du paramètre "Pourcentage de risque".

```
void  Percent(
   double  percent      // pourcentage de risque
   )
```

Paramètres

percent

[in] Pourcentage de risque.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La méthode ValidationSettings() de la classe de base retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Retourne le volume d'une position longue.

```
virtual double  CheckOpenLong(
   double  price,      // prix
   double  sl          // Stop Loss
   )
```

Paramètres

price

[in] Prix d'ouverture d'une position longue.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Retourne le volume d'une position courte.

```
virtual double  CheckOpenShort(
   double  price,      // prix
   double  sl          // Stop Loss
   )
```

Paramètres

price

[in] Prix d'ouverture d'une position courte.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position courte.

© 2000-2024, MetaQuotes Ltd.

CheckReverse

Retourne le volume pour renverser une position.

```
virtual double  CheckReverse(
   CPositionInfo*  position,      // pointeur
   double          sl             // Stop Loss
   )
```

Paramètres

position

[in] Pointeur sur un objet de classe CPositionInfo.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume pour renverser la position.

© 2000-2024, MetaQuotes Ltd.

CheckClose

Vérifie les conditions pour clôturer une position ouverte.

```
virtual double  CheckClose()
```

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Modules des Signaux de Trading

Le terminal client inclut en standard un ensemble de modules de signaux de trading prêts-à-l'emploi pour l'"Assistant MQL5". Lors de la création d'un Expert Advisor avec l'Assistant MQL5, vous pouvez utilisez n'importe quelle combinaison des modules des signaux de trading (jusqu'à 64). La décision finale d'une opération de trading est faite sur la base d'une analyse complexe des singaux obtenus de tous les modules inclus. La description détaillée du mécanisme de la prise de décision est donnée ci-dessous.

Les modules de signaux suivants sont disponibles en standard :

- Signaux de l'indicateur Oscillateur d'Accélération
- Signaux de l'Indicateur Moyenne Mobile Adaptative
- Signaux de l'indicateur Awesome Oscillator
- Signaux de l'indicateur Oscillateur Bears Power
- Signaux de l'indicateur Oscillateur Bulls Power
- Signaux de l'indicateur Oscillateur Commodity Channel Index
- Signaux de l'indicateur Oscillateur DeMarker
- Signaux de l'indicateur Double Exponential Moving Average
- Signaux de l'indicateur Envelopes
- Signaux de l'indicateur Fractal Adaptive Moving Average
- Signaux du Filtre Horaire Intraday
- Signaux de l'oscillateur MACD
- Signaux de l'indicateur Moving Average
- Signaux de l'indicateur Parabolic SAR
- Signaux de l'oscillateur Relative Strength Index
- Signaux de l'oscillateur Relative Vigor Index
- Signaux de l'oscillateur Stochastic
- Signaux de l'oscillateur Triple Exponential Average
- Signaux de l'indicateur Triple Exponential Moving Average
- Signaux de l'oscillatuer Williams Percent Range

Mécanisme de Prise de Décision sur la Base des Modules de Signaux

Le mécanisme de prise de décision peut être représenté comme la liste suivante des principes de base :

- Chaque module de signal a ses propres modules de marché (une combinaison de prix et de valeurs d'un indicateur).
- Chaque modèle de marché a une valeur qui peut varier dans l'intervalle 1 à 100. Plus cette valeur est haute, plus le modèle est fort.
- Chaque modèle génère une prévision de la direction du mouvement des prix.
- La prévision d'un module est le résultat de la recherche de modèles intégrés, et est représentée comme un chiffre dans l'intervalle -100 à 100. Le signe détermine la direction de la prévision du

© 2000-2024, MetaQuotes Ltd.

mouvement (un signe négatif signifie que les prix vont baisser, un chiffre positif que les prix vont monter). La valeur absolue correspond à la puissance du meilleur modèle trouvé.

- La prévision de chaque module est envoyé pour un "vote" final avec un coefficient de 0 à 1 spécifié dans ses paramètres (paramètre "Weight").
- Le résultat du vote est un nombre dans l'intervalle -100 à 100, où le signe détermine la direction de la prévision du mouvement, et la valeur absolue caractérise la puissance du signal. Il est calculé comme étant la moyenne arithmétique des prévisions pondétées de tous les modules de signaux.

Chaque Expert Advisor généré a deux paramètres modifiables — les niveaux de déclenchement d'ouverture et de clôture d'une position (ThresholdOpen et ThresholdClose) qui peuvent être égaux à une valeur de l'intervalle 0 à 100. Si la puissance du signal final est supérieur au niveau de déclenchement, une opération de trading correspondant au signe du signal est effectuée.

Exemples

Considérons un Expert Advisor avec les niveaux de déclenchement suivants : ThresholdOpen=20 et ThresholdClose=90. Deux modules de signaux participent à la prise de décision des opérations de trading : le module MA avec un poids de 0,4 et le module Stochastic avec un poids de 0,8. Analysons deux possibilités de signaux de trading obtenus :

Possibilité 1.

Le prix a croisé la Moyenne Mobile (MA) haussière à la hausse. Ce cas correspond à un des modèles de marché implémenté dans le module MA. Ce modèle implique une hausse des prix. Sa valeur est égale à 100. En même temps, l'oscillateur Stochastic s'est retourné à la baisse et forme une divergence avec les prix. Ce cas correspond à un des modèle de marché implémenté dans le module Stochastic. Ce modèle implique une baisse des prix. Le poids de ce modèle est 80.

Calculons le résult final du "vote". La valeur obtenue du module MA est 0,4 * 100 = 40. La valeur du module Stochastic est 0,8 * (-80) = -64. La valeur finale est la moyenne arithmétique de ces 2 valeurs : (40 - 64)/2 = -12. Le résultat du vote est un signal de vente avec une force relative de 12. Le niveau de déclenchement qui est à 20 n'est pas atteint. Aucune opération de trading ne sera donc déclenchée.

Possibilité 2.

Le prix a croisé la Moyenne Mobile (MA) haussière à la baisse. Ce cas correspond à un des modèles implémenté dans le module MA. Ce modèle implique une hausse des prix. Sa valeur est égale à 10. En même temps, l'oscillateur Stochastic s'est retourné à la baisse et forme une divergence avec les prix. Ce cas correspond à un des modèle de marché implémenté dans le module Stochastic. Ce modèle implique une baisse des prix. Le poids de ce modèle est 80.

Calculons le résult final du "vote". La valeur obtenue du module MA est 0,4 * 10 = 4. La valeur du module Stochastic est 0,8 * (-80) = -64. La valeur finale est la moyenne arithmétique de ces 2 valeurs : (4 - 64)/2 = -30. Le résultat du vote est un signal de vente avec une force relative de 30. Le niveau de déclenchement qui est à 20 est atteint. Le résultat est l'obtention d'un signal d'ouverture d'une position courte.

© 2000-2024, MetaQuotes Ltd.

a) Divergence des prix avec l'oscillateur Stochastic (possibilités 1 et 2).

b) Les prix ont croisé la Moyenne Mobile à la hausse (possibilité 1).

c) Les prix ont croisé la Moyenne Mobile à la baisse (possibilité 2).

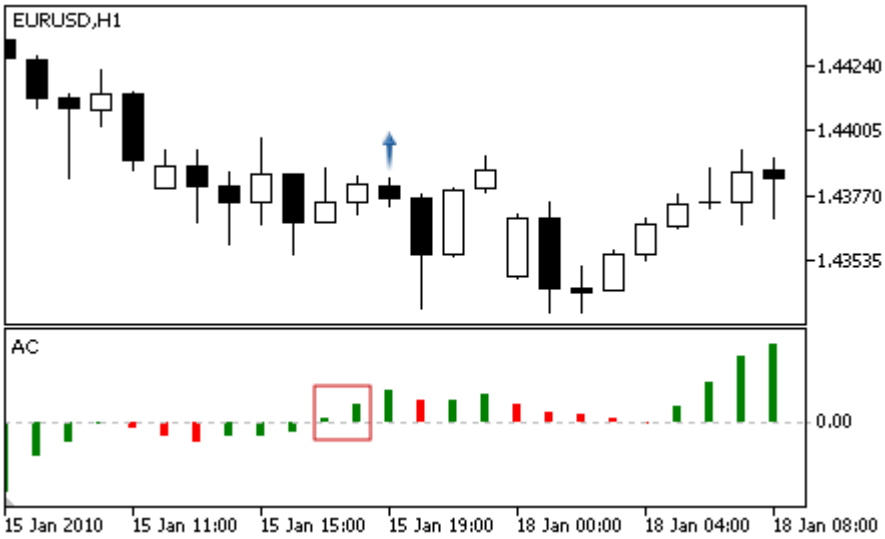
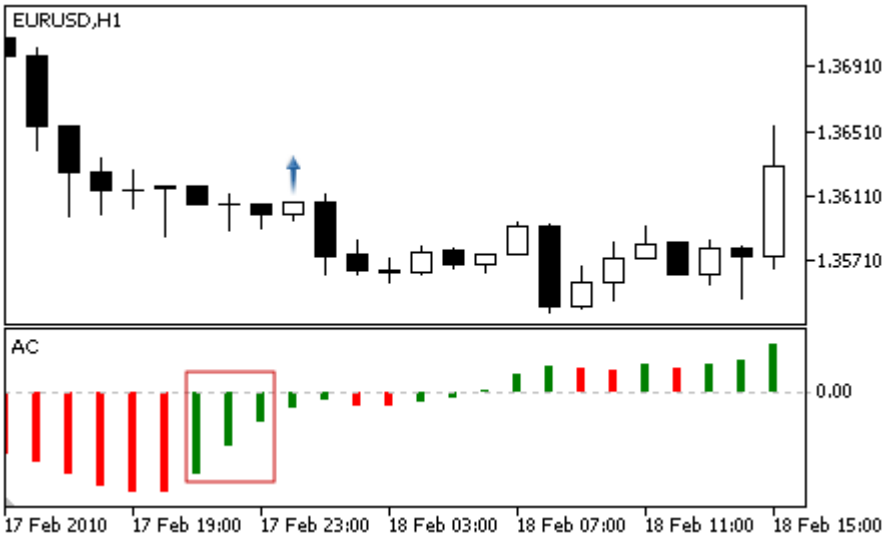
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Oscillateur d'Accélération

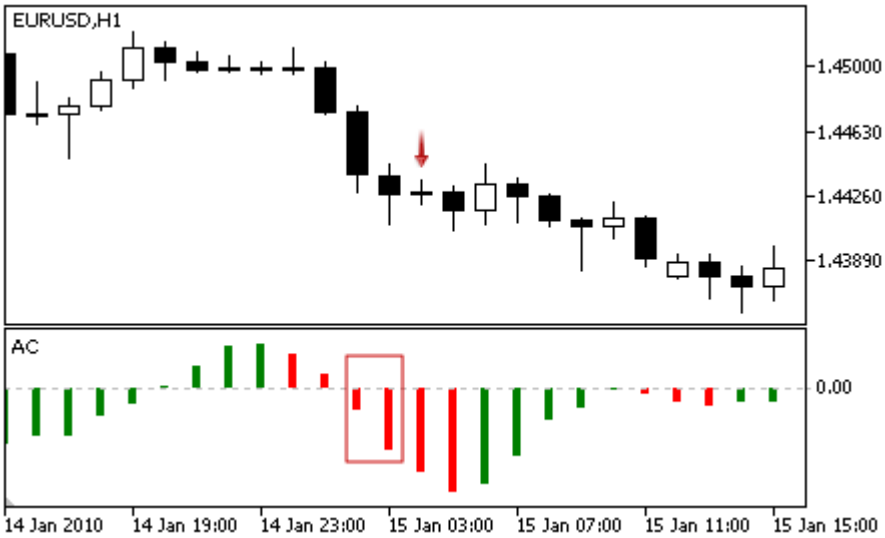
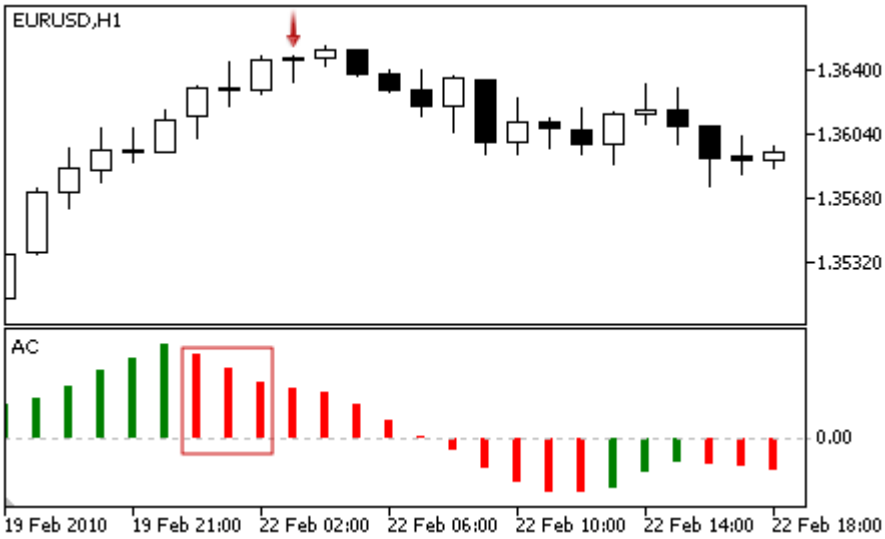
Ce module est basé sur les modèles de marché de l'indicateur Oscillateur d'Accélération. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• La valeur de l'indicateur est supérieure à 0 et augmente sur la barre analysée et la barre précédente.  • La valeur de l'indicateur est inférieure à 0 et augmente sur la barre analysée et la barre précédente. 
For selling	• La valeur de l'indicateur est inférieure à 0 et diminue sur la barre analysée et la barre précédente.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • La valeur de l'indicateur est inférieure à 0 et diminue sur la barre analysée et la barre précédente. 
Aucune objection pour l'achat	La valeur de l'indicateur augmente sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.

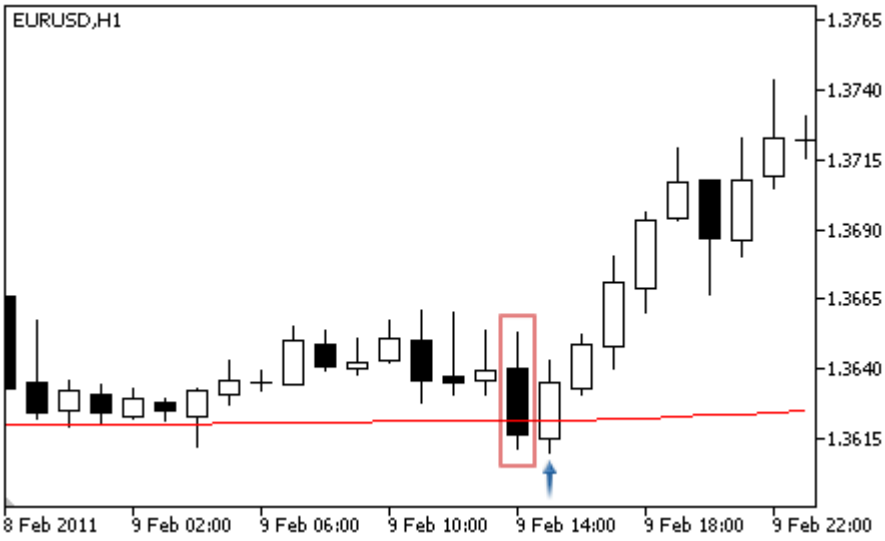
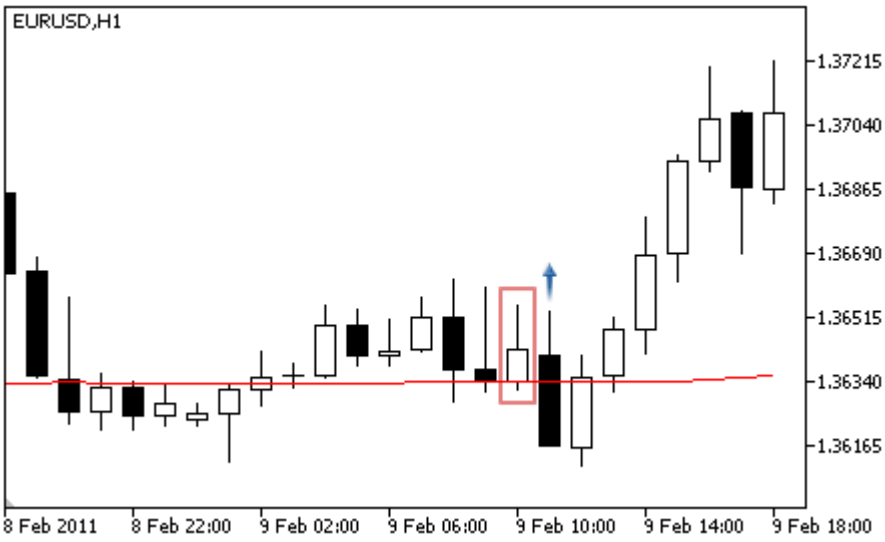
© 2000-2024, MetaQuotes Ltd.

Signaux de l'Indicateur Moyenne Mobile Adaptative

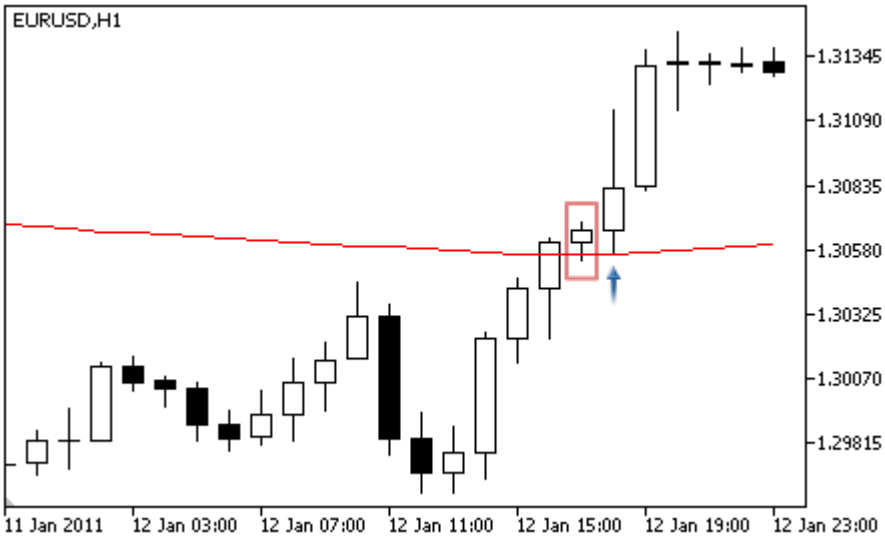
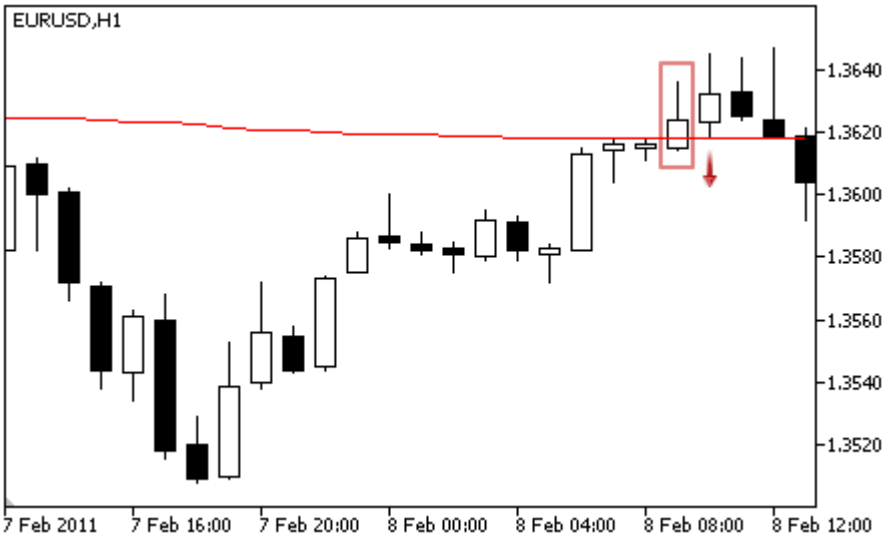
Ce module est basé sur les modèles de marché de l'indicateur Adaptive Moving Average (Moyenne Mobile Adaptative). Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

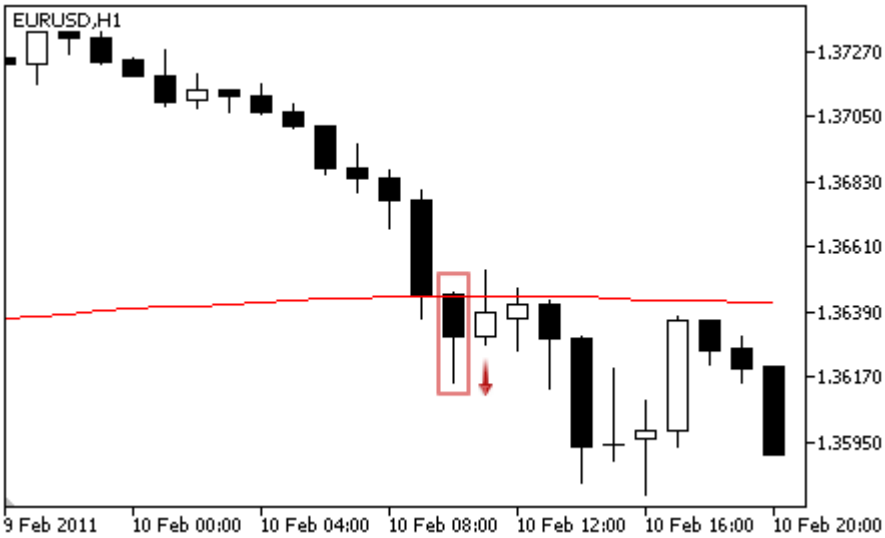
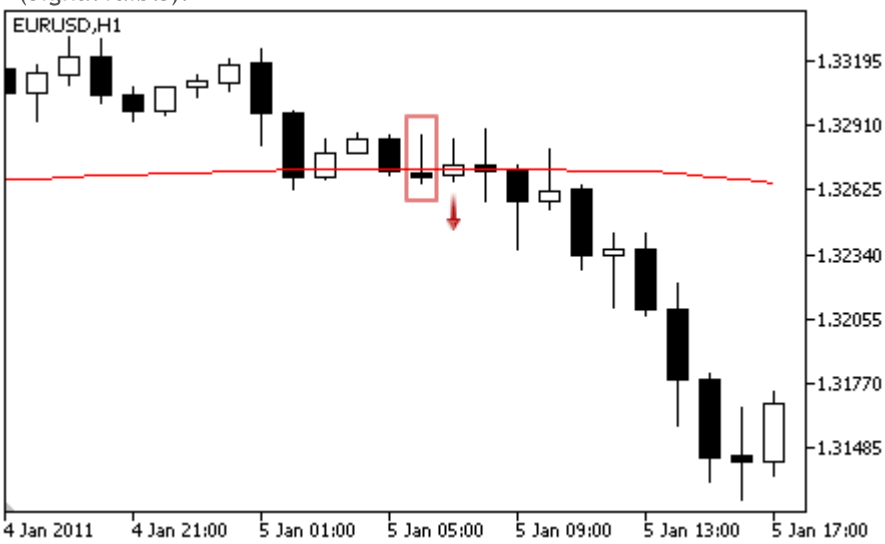
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en hausse (signal fort). 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• L'ombre inférieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont au-dessus de l'indicateur, et le prix le plus bas est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible). 
For selling	• Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en baisse (signal fort).

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • L'ombre supérieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont en-dessous de l'indicateur, et le prix le plus haut est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible). 
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de lissage de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.

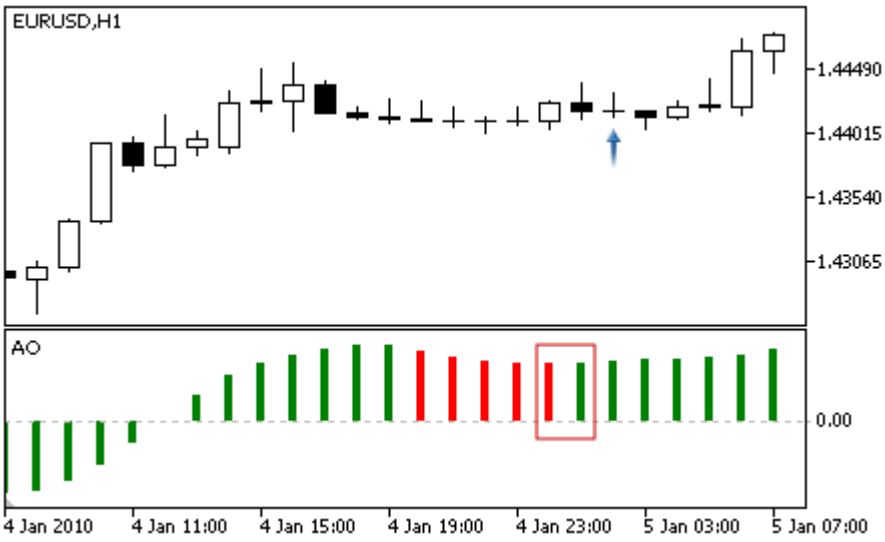
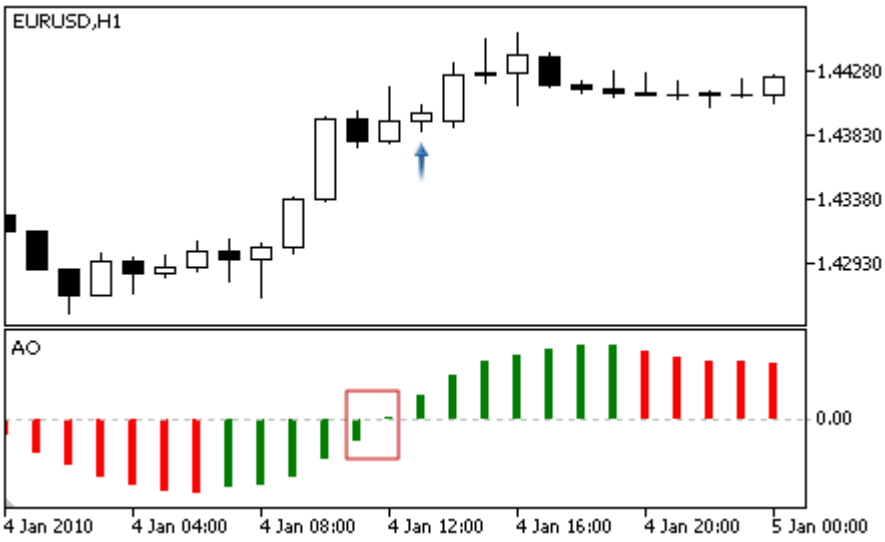
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Awesome Oscillator

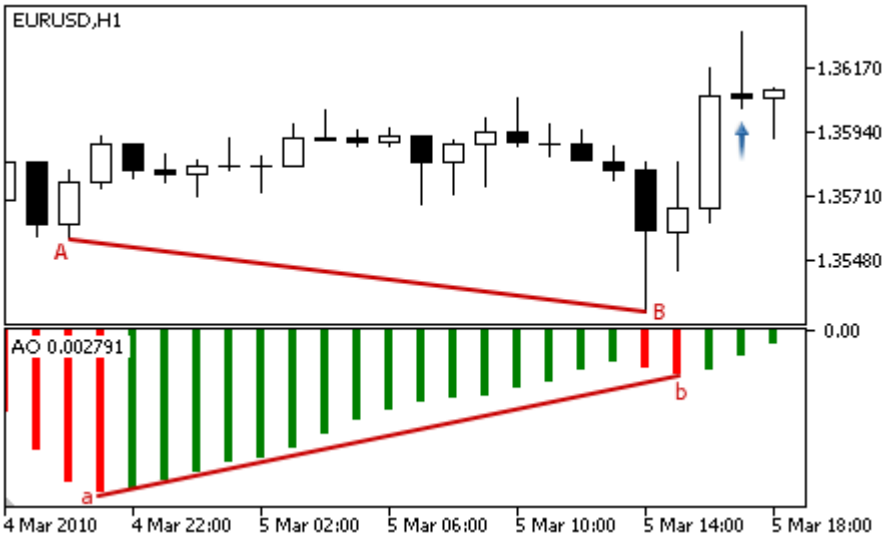
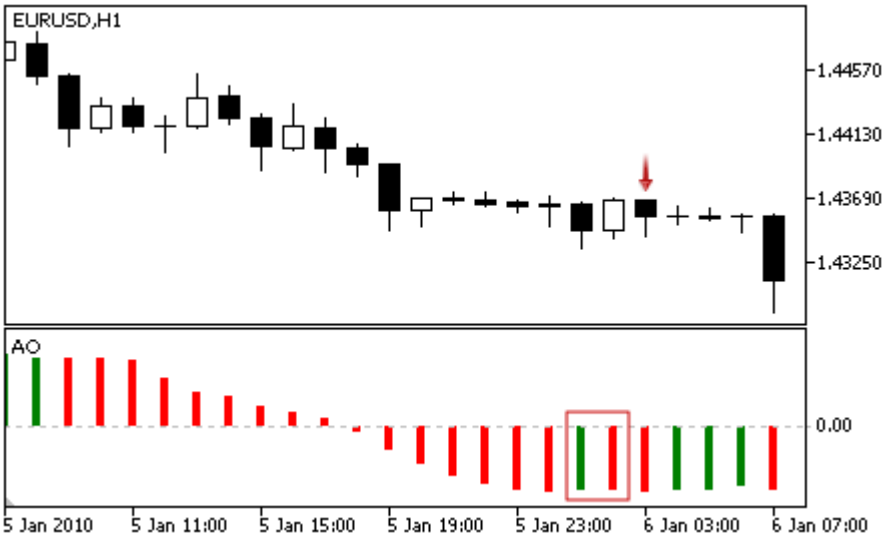
Le module des signaux basés sur les modèles de marché de l'indicateur Awesome Oscillator. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

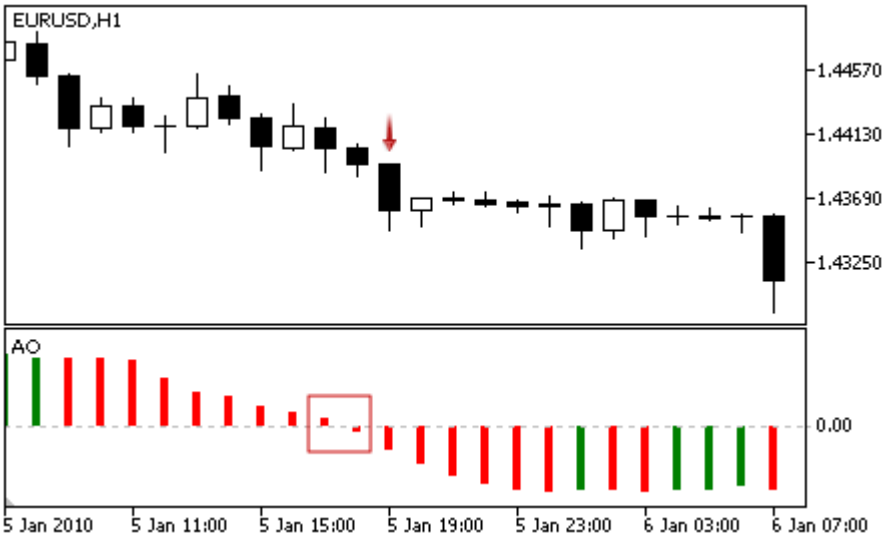
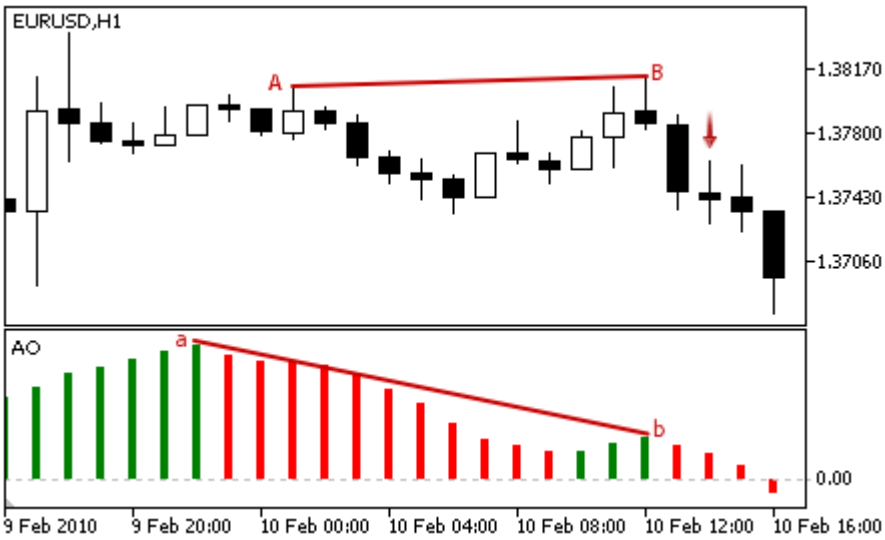
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Saucer** — la valeur de l'indicateur est en hausse sur la barre en cours d'analyse, alors qu'elle baissait sur la barre précédente ; les deux valeurs sont supérieure à 0.  • **Croisement de la ligne zéro** — la valeur de l'indicateur est positive sur la barre en cours d'analyse, et inférieure à 0 sur la barre précédente. 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• **Divergence** — le premier creux analysé de l'indicateur est moins profond que le précédent, et le creux correspondant des prix est plus profond que le précédent. De plus, l'indicateur ne doit pas repasser au-dessus de 0. 
For selling	• **Saucer** — la valeur de l'indicateur est en baisse sur la barre en cours d'analyse et grimpait sur la barre précédente, les deux valeurs sont inférieures à 0.  • **Croisement de la ligne 0** — la valeur de l'indicateur est inférieure à 0 sur la barre en cours d'analyse, et était supérieure à 0 sur la barre précédente.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Divergence** — le premier pic analysé de l'indicateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le précédent. De plus, l'indicateur ne doit pas passer sous la barre zéro. 
Aucune objection pour l'achat	La valeur de l'indicateur augmente sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.

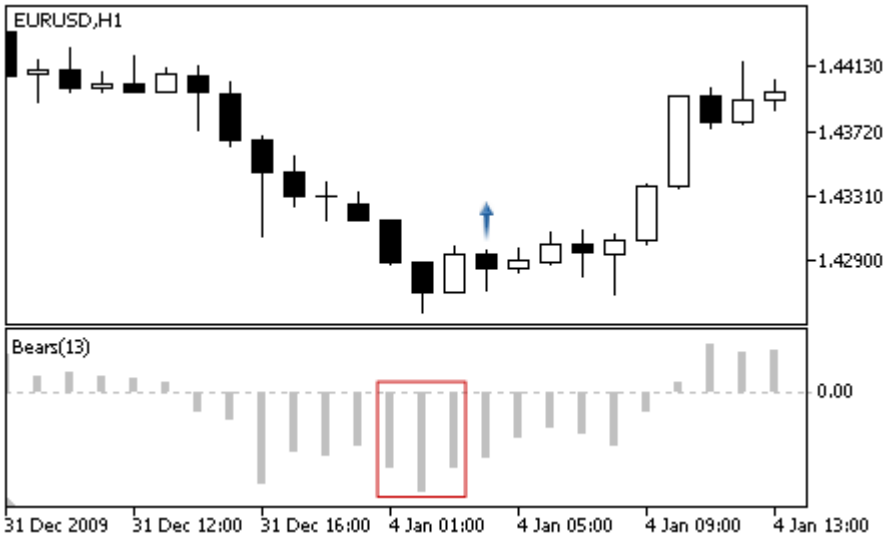
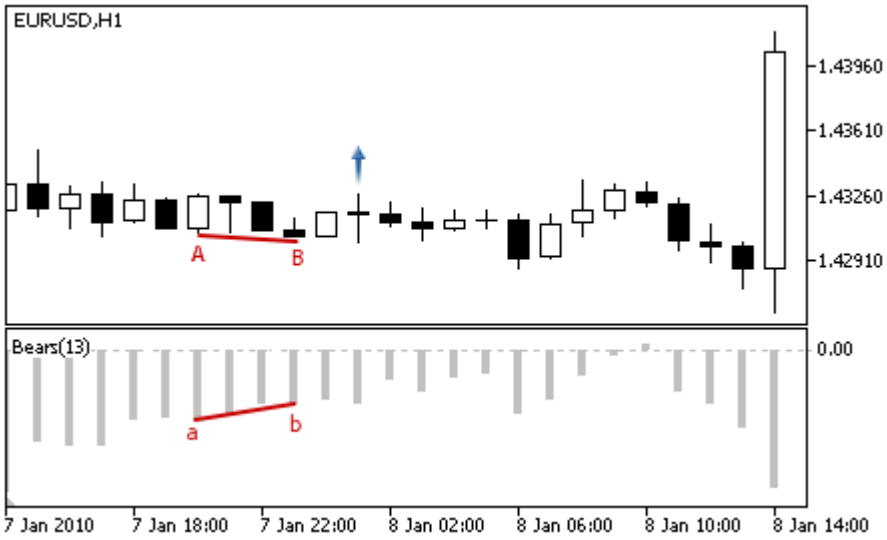
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Oscillateur Bears Power

Le module des signaux basés sur les modèles de marché de l'oscillateur Bears Power. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Renversement** — l'oscillateur s'est retourné à la hausse et sa valeur sur la barre en cours d'analyse est inférieure à 0.  • **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent. De plus, l'oscillateur ne doit grimper au-dessus de 0. 
For selling	Aucun signal de vente.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
Aucune objection pour l'achat	La valeur de l'oscillateur est inférieure à 0.
Aucune objection pour la vente	Aucun signal.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodBears	Période de calcul de l'oscillateur.

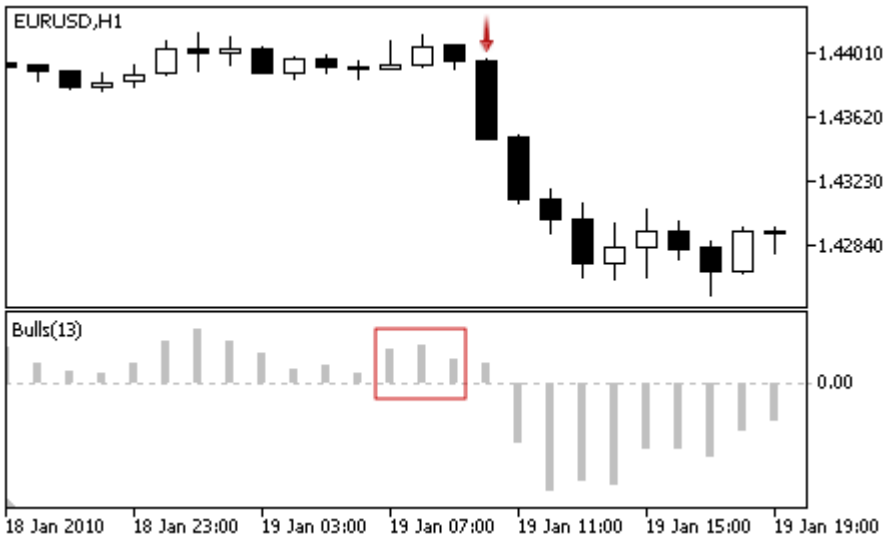
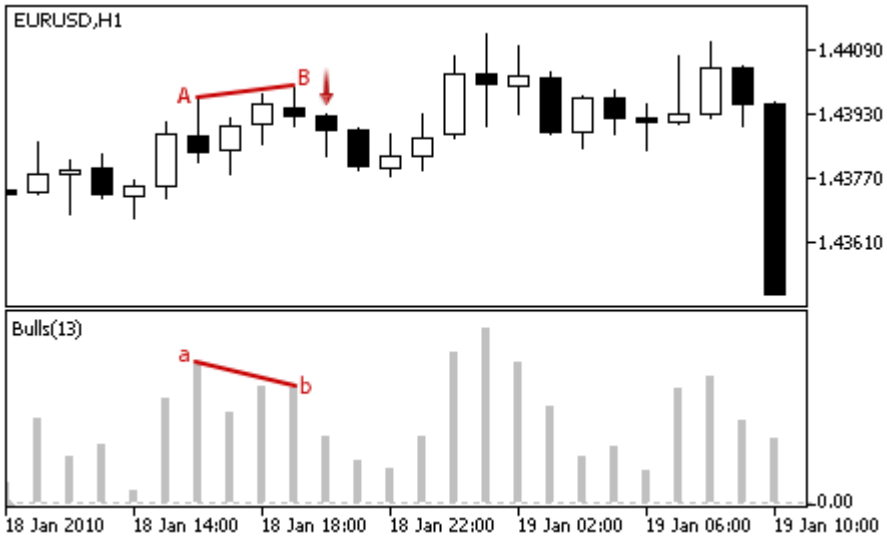
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Oscillateur Bulls Power

Ce module de signaux est basé sur les modèles de marché de l'oscillateur Bulls Power. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	Aucun signal d'achat.
For selling	• **Renversement** — l'oscillateur s'est retourné à la baisse et sa valeur sur la barre en cours d'analyse est supérieure à 0.  • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent. De plus, l'oscillateur ne doit pas chuter en dessous de 0. 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
Aucune objection pour l'achat	Aucun signal.
Aucune objection pour la vente	La valeur de l'oscillateur est supérieure à 0.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodBulls	Période de calcul de l'oscillateur.

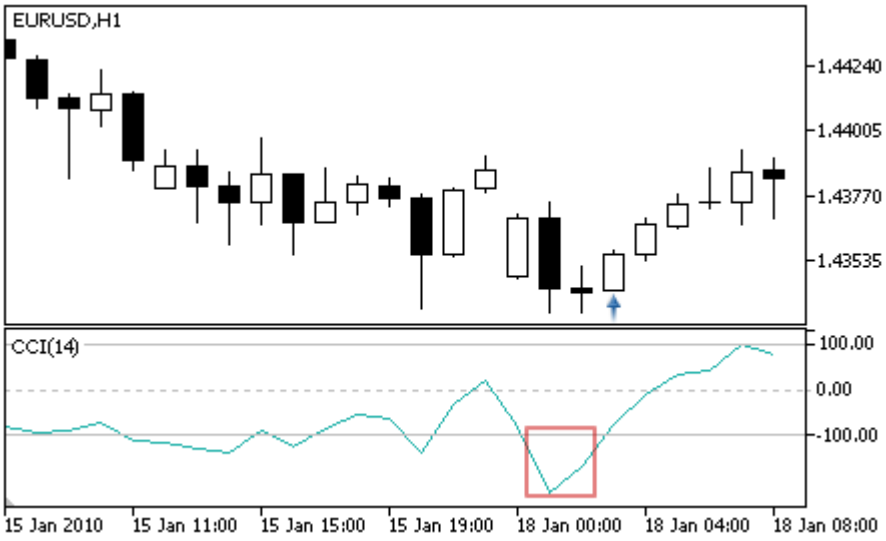
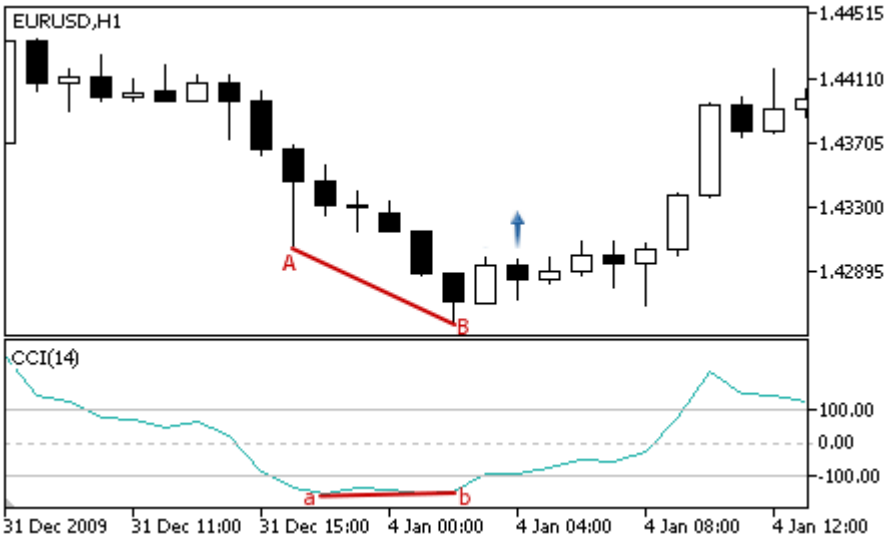
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Oscillateur Commodity Channel Index

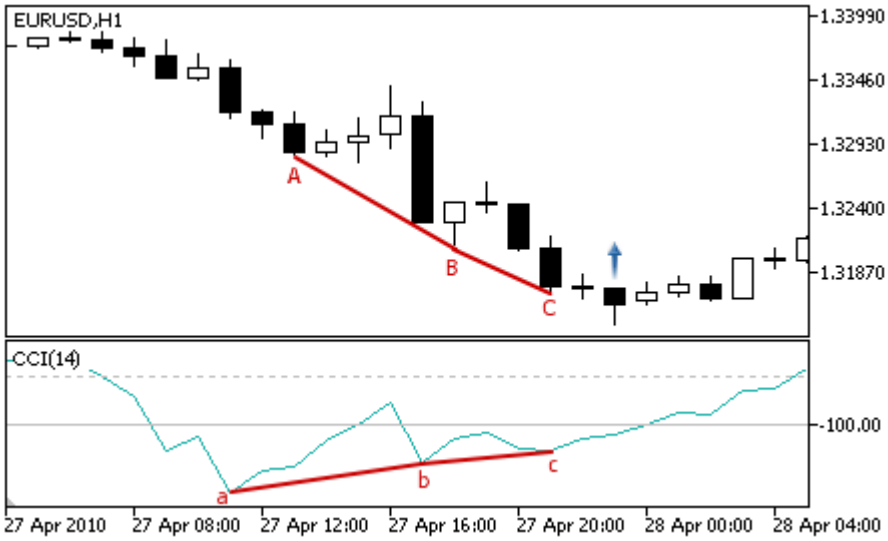
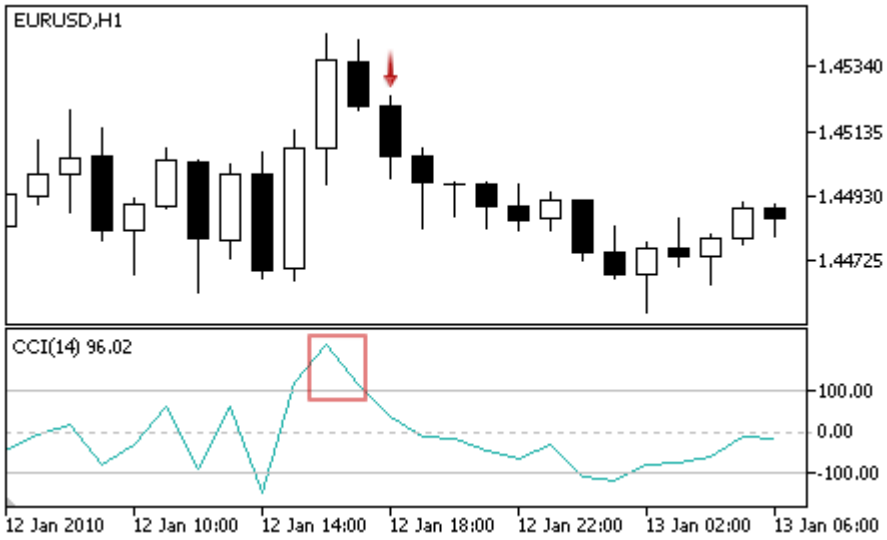
Ce module de signaux est basé sur les modèles de marché de l'oscillateur Commodity Channel Index. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

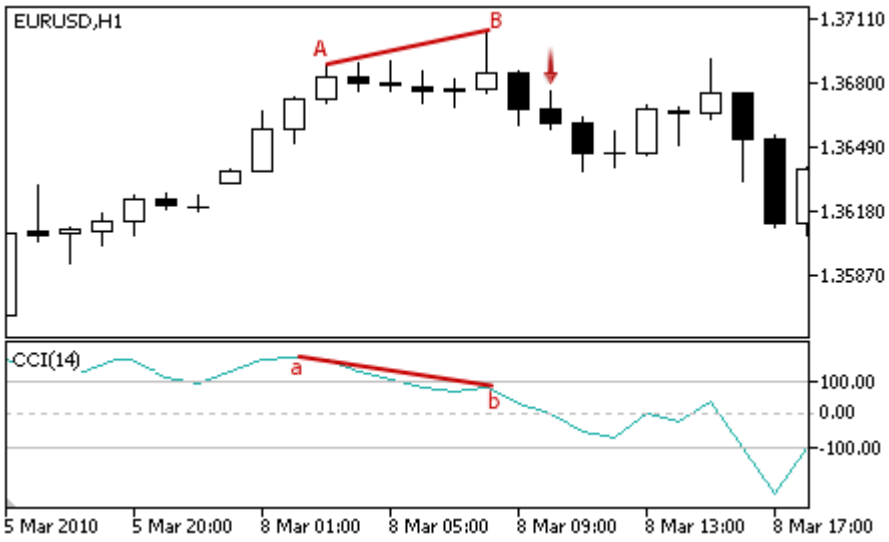
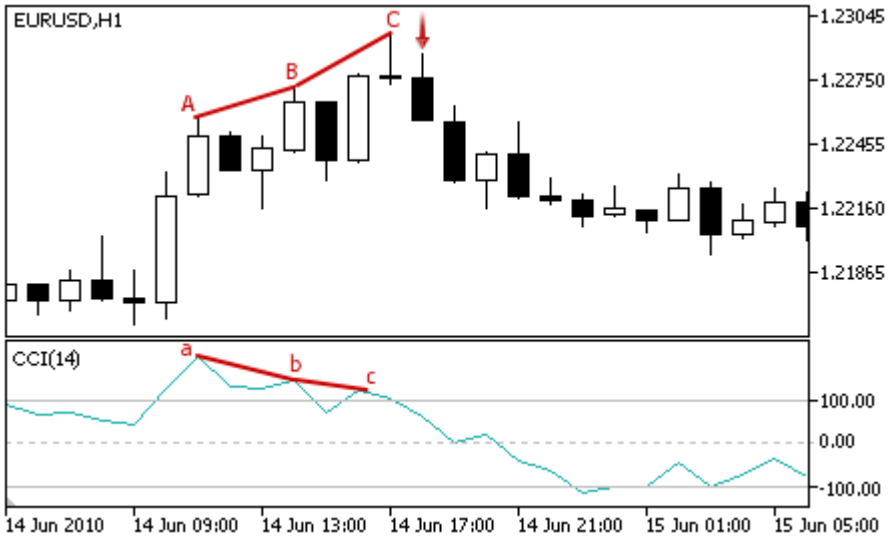
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Renversement en zone de sur-vente** — l'oscillateur se retourne à la hausse et sa valeur sur la barre en cours d'analyse est dans la zone de sur-vente (la valeur par défaut est -100).  • **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent. 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• **Double Divergence** — l'oscillateur forme 3 creux consécutifs, chacun d'eux est supérieur au précédent ; et les prix forment également les 3 creux correspondants, et chacun d'eux est plus bas que le précédent. 
For selling	• **Renversement en zone de sur-achat** — l'oscillateur se retourne à la baisse et sa valeur sur la barre en cours d'analyse est dans la zone de sur-achat (la valeur par défaut est 100).  • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Double Divergence** — l'oscillateur forme 3 pics consécutifs, chacun d'eux est inférieur au précédent ; et les prix forment également les 3 pics correspondants, et chacun d'eux est plus haut que le précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodCCI	Période de calcul de l'oscillateur.
Applied	Une série de prix utilisée pour le calcul de l'oscillateur.

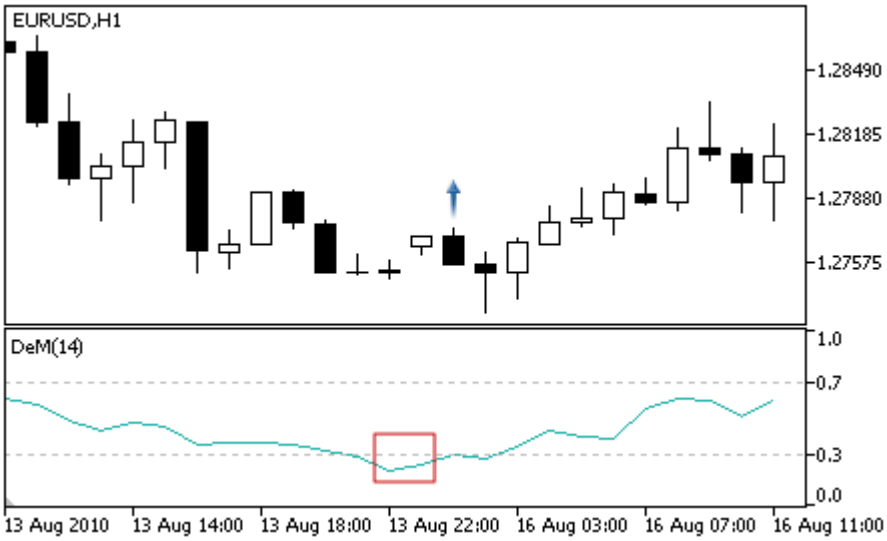
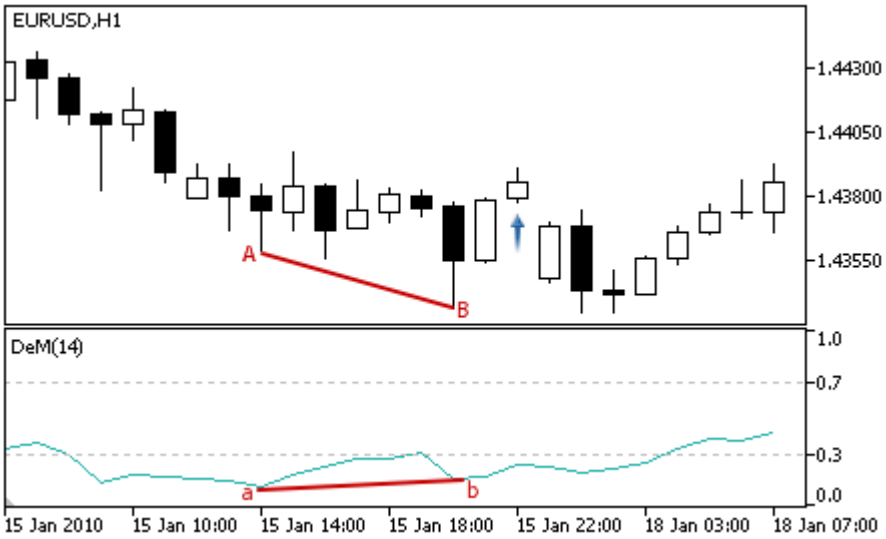
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Oscillateur DeMarker

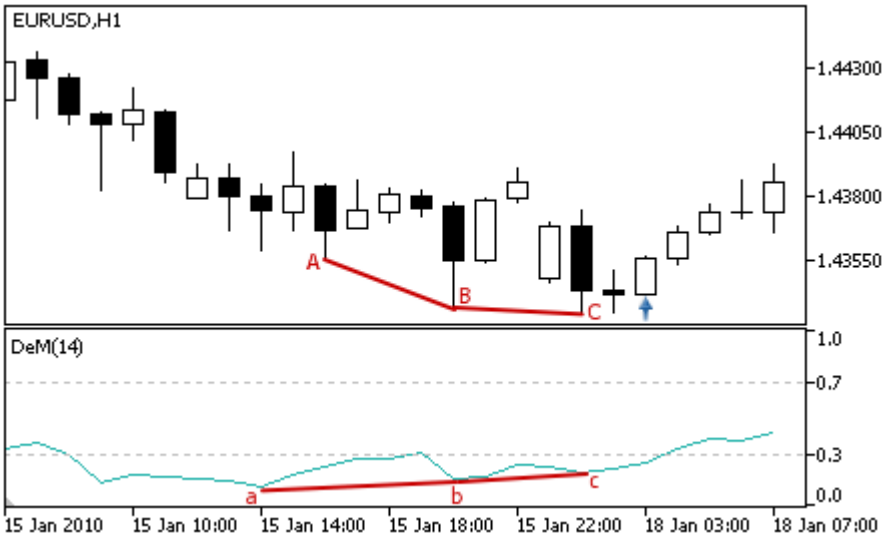
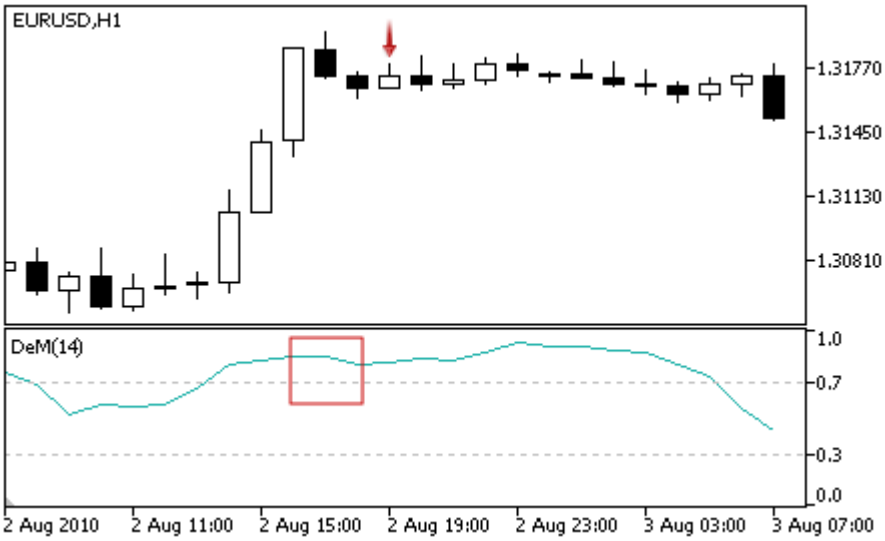
Ce module de signaux est basé sur les modèles de marché de l'oscillateur DeMarker. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

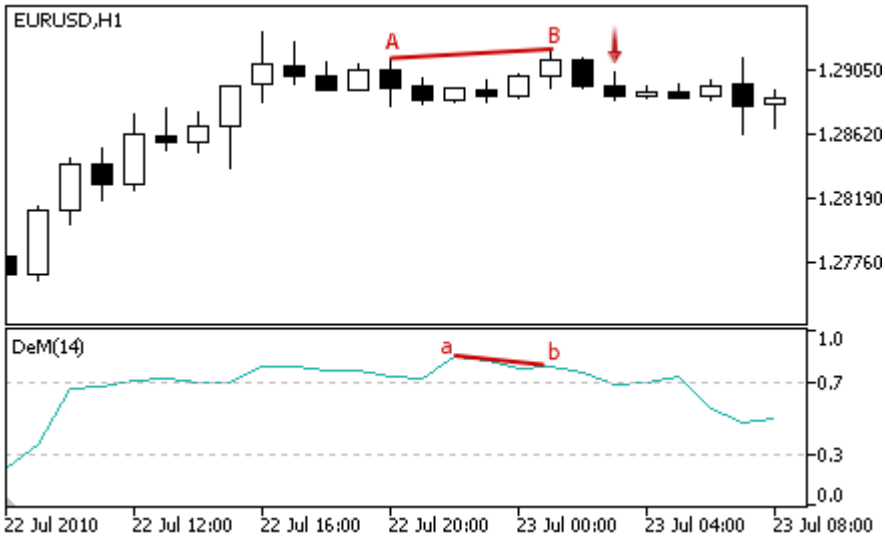
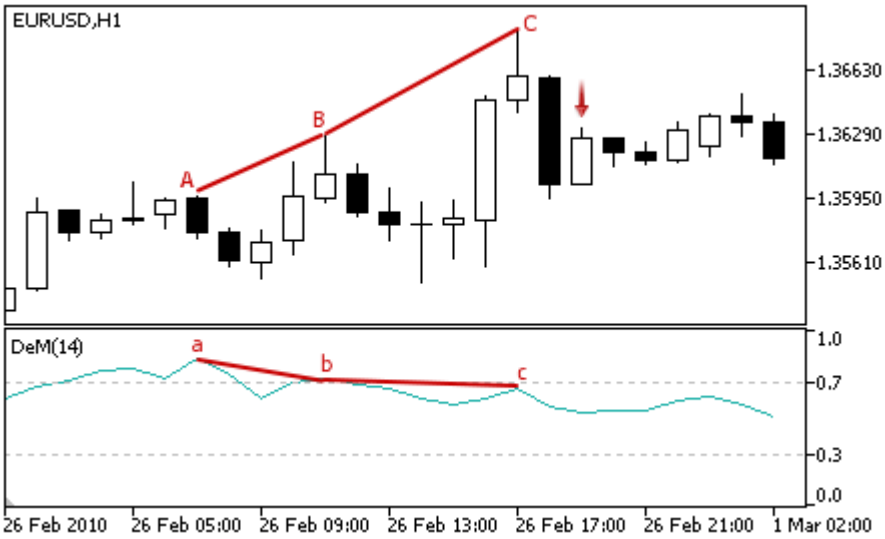
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Retournement dans la zone de sur-vente** — l'oscillateur se retourne à la hausse et sa valeur sur la barre analysée est dans la zone de sur-vente (la valeur par défaut est 0,3).  • **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent. 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• **Double Divergence** — l'oscillateur forme 3 creux consécutifs, chacun d'eux est supérieur au précédent ; et les prix forment également les 3 creux correspondants, et chacun d'eux est plus bas que le précédent. 
For selling	• **Renversement en zone de sur-achat** — l'oscillateur se retourne à la baisse et sa valeur sur la barre en cours d'analyse est dans la zone de sur-achat (la valeur par défaut est 0,7).  • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Double Divergence** — l'oscillateur forme 3 pics consécutifs, chacun d'eux est inférieur au précédent ; et les prix forment également les 3 pics correspondants, et chacun d'eux est plus haut que le précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodDeM	Période de calcul de l'oscillateur.

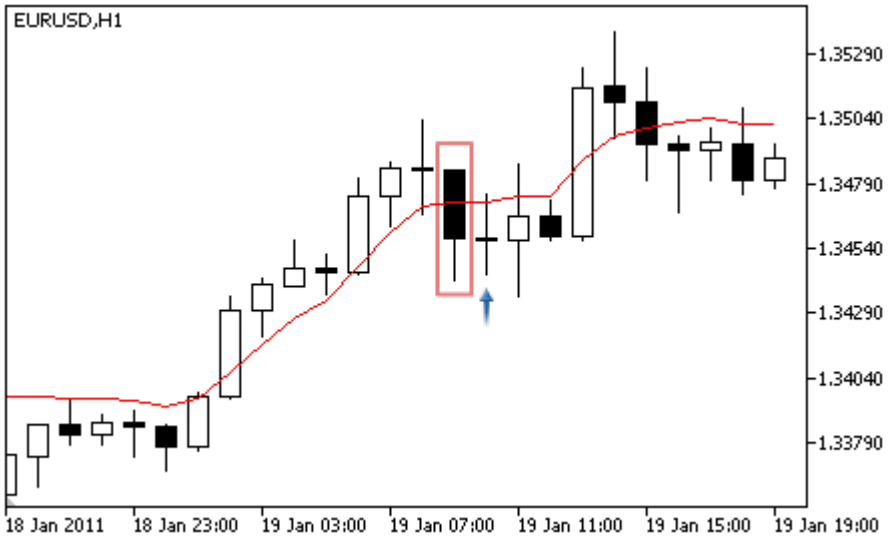
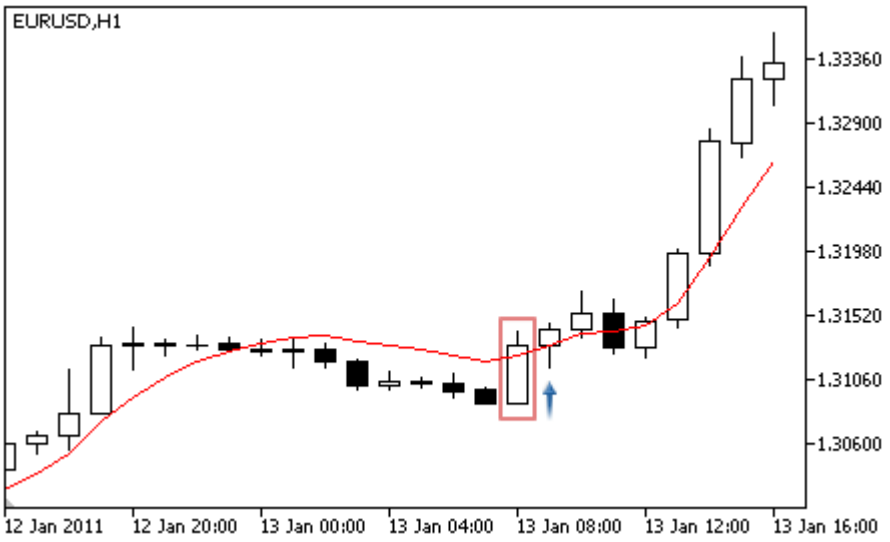
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Double Exponential Moving Average

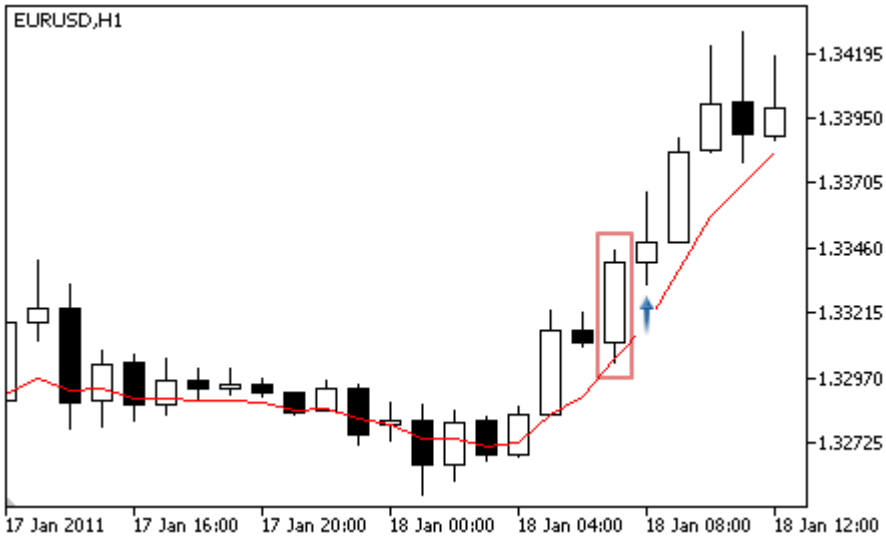
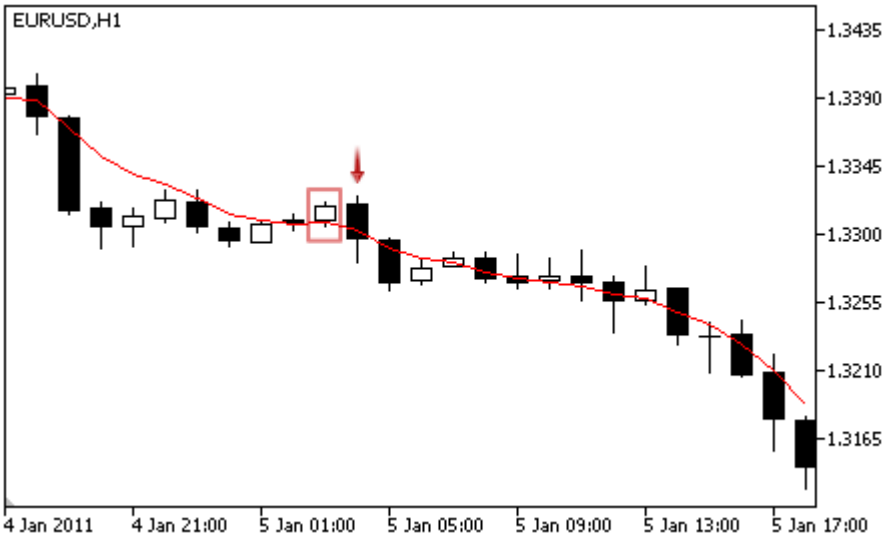
Ce module est basé sur les modèles de marché de l'indicateur >Double Exponential Moving Average (Moyenne Mobile Exponentielle Double). Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

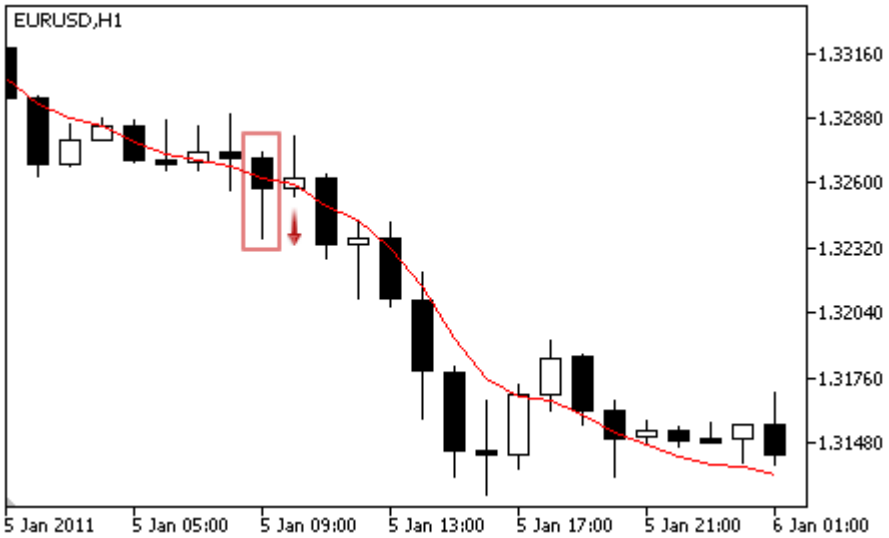
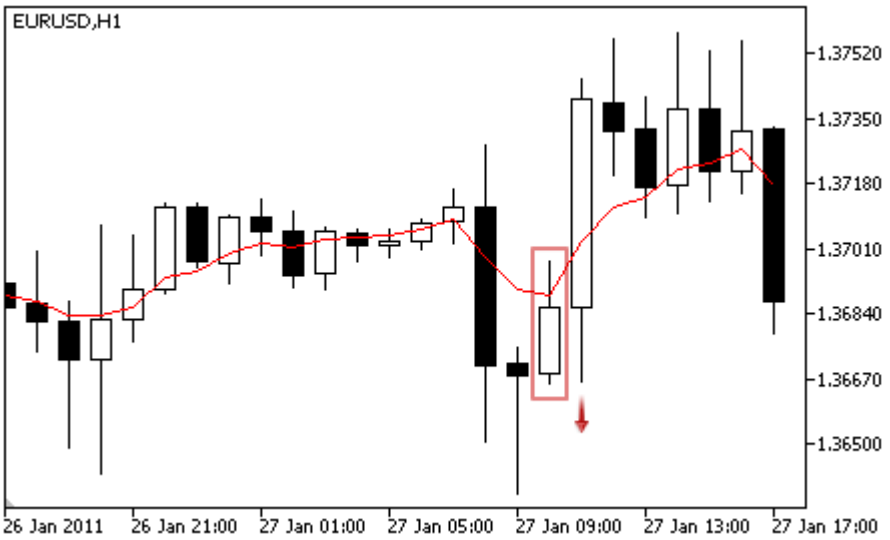
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en hausse (signal fort). 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• L'ombre inférieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont au-dessus de l'indicateur, et le prix le plus bas est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible). 
For selling	• Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en baisse (signal fort).

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • L'ombre supérieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont en-dessous de l'indicateur, et le prix le plus haut est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible). 
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de lissage de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.

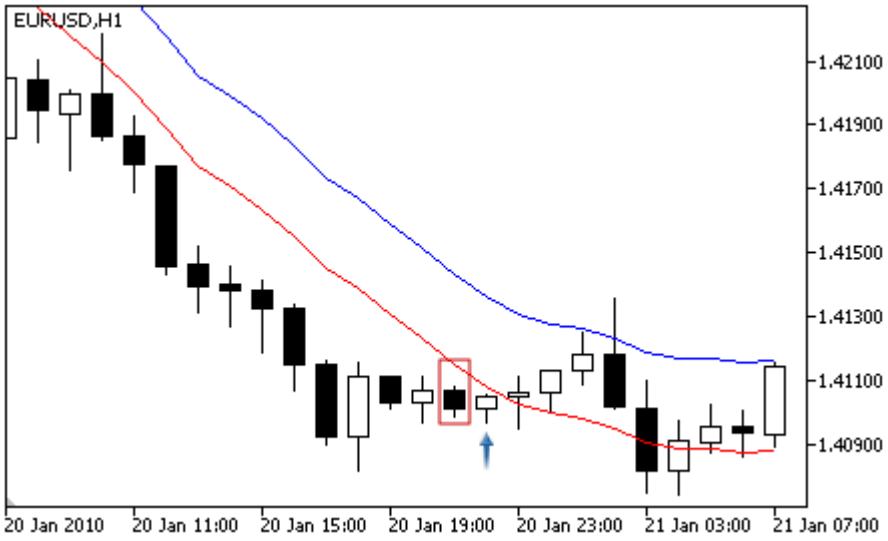
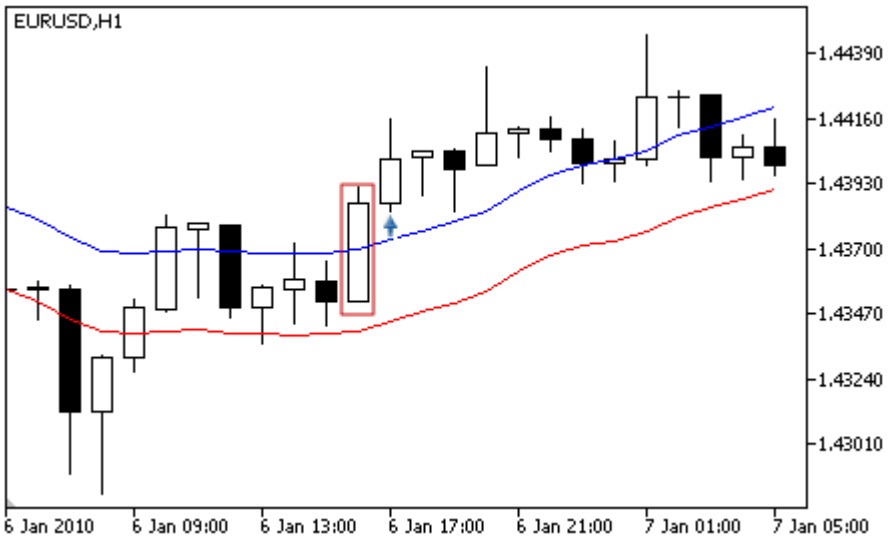
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Envelopes

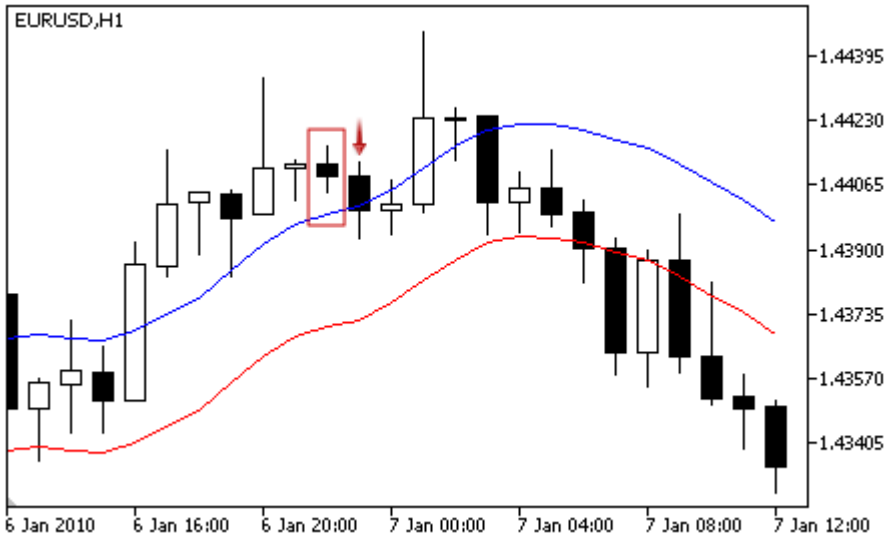
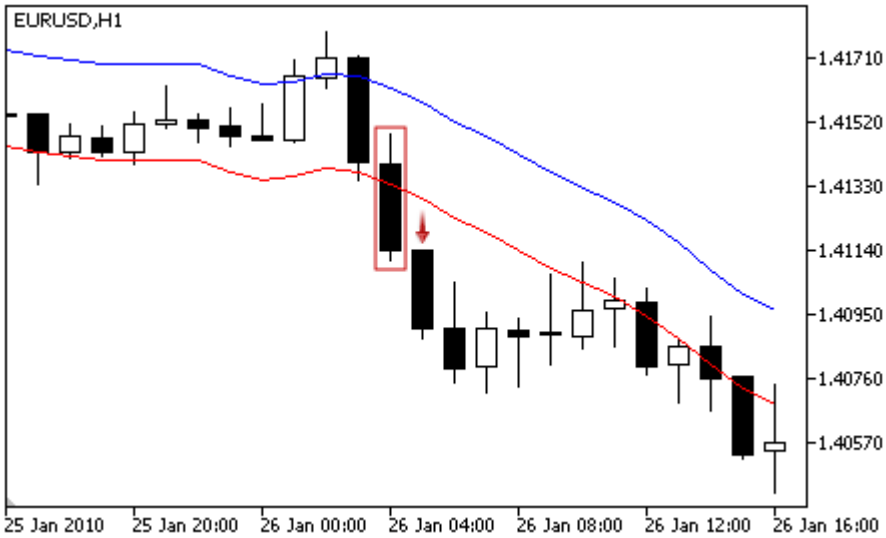
Ce module de signaux est basé sur les modèles de marché de l'indicateur Awesome Envelopes. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• Le prix est près de la ligne inférieure de l'indicateur sur la barre analysée.  • Le prix a croisé la ligne supérieure de l'indicateur sur la barre analysée. 
For selling	• Le prix est près de la ligne supérieure de l'indicateur sur la barre analysée.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • Le prix a croisé la ligne inférieure de l'indicateur sur la barre analysée. 
Aucune objection pour l'achat	Aucun signal.
Aucune objection pour la vente	Aucun signal.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de calcul de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.
Deviation	Déviation des bordures de l'enveloppe depuis la ligne centrale (la moyenne mobile) en pourcentages.

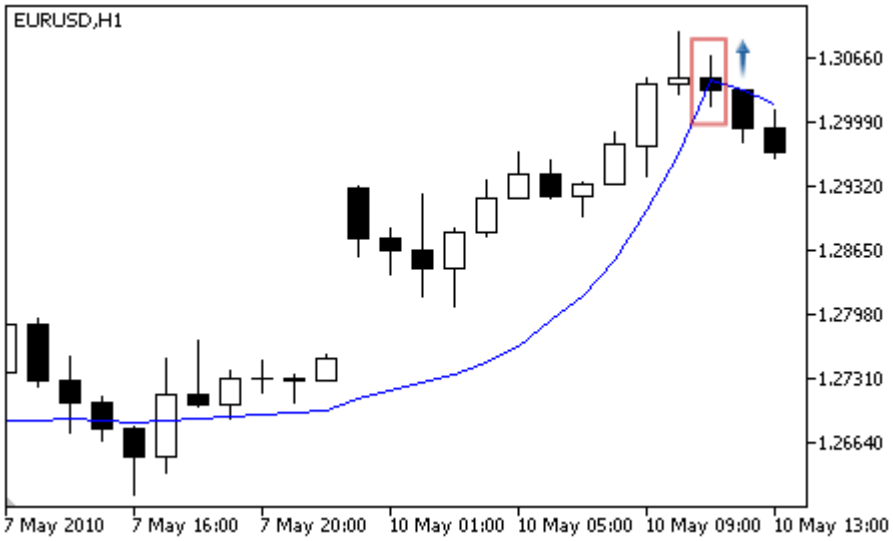
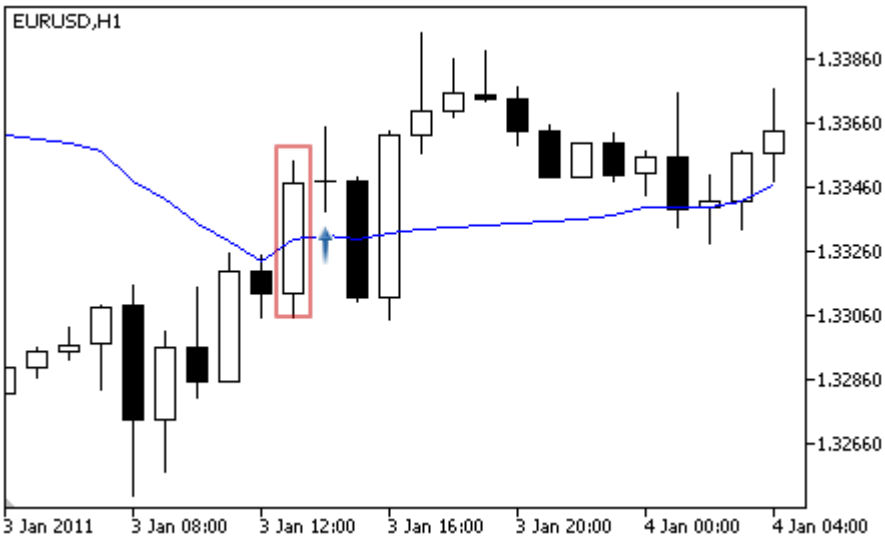
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Fractal Adaptive Moving Average

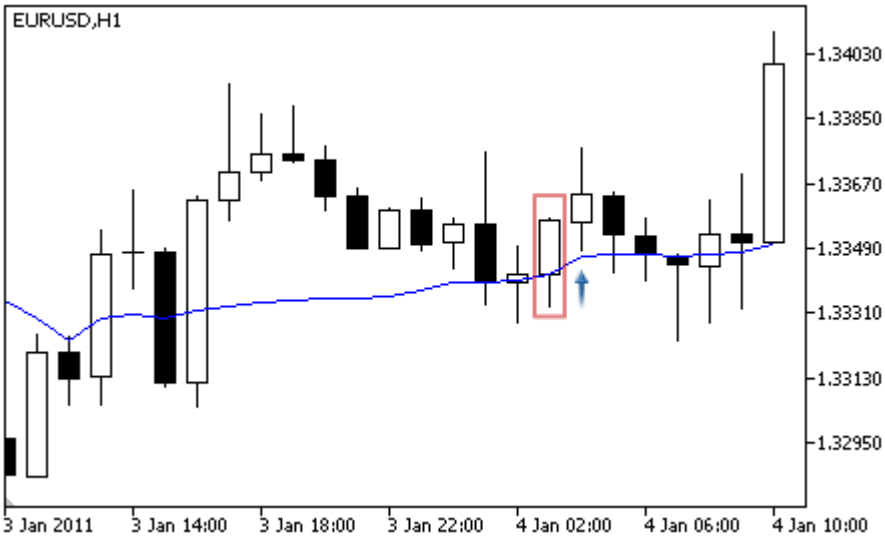
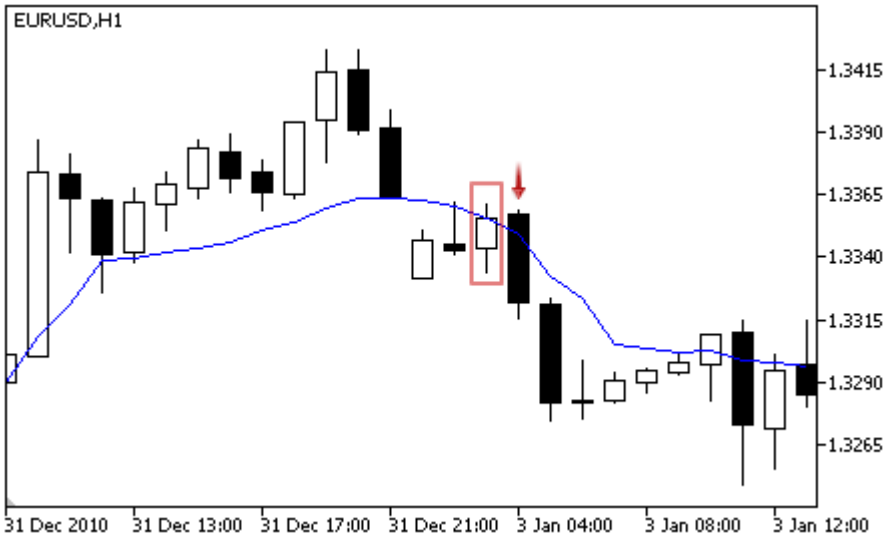
Ce module de signaux est basé sur les modèles de marché de l'indicateurFractal Adaptive Moving Average. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

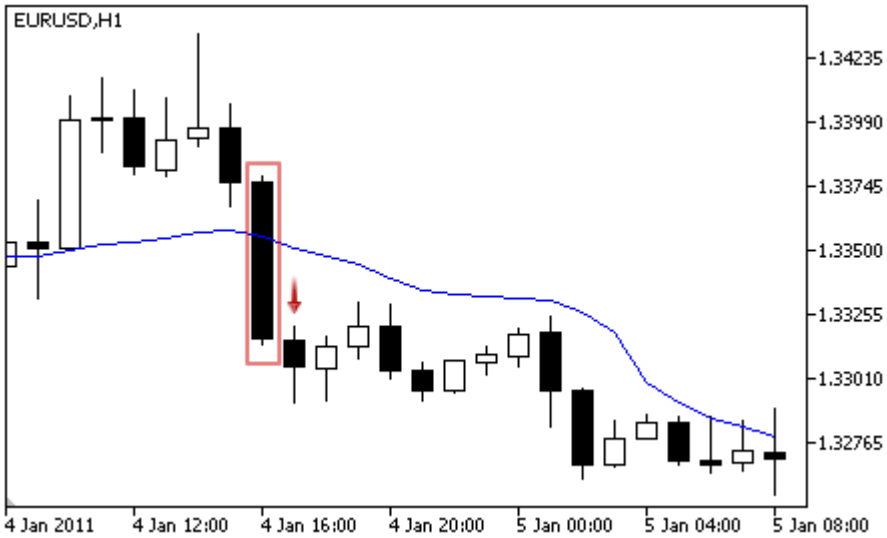
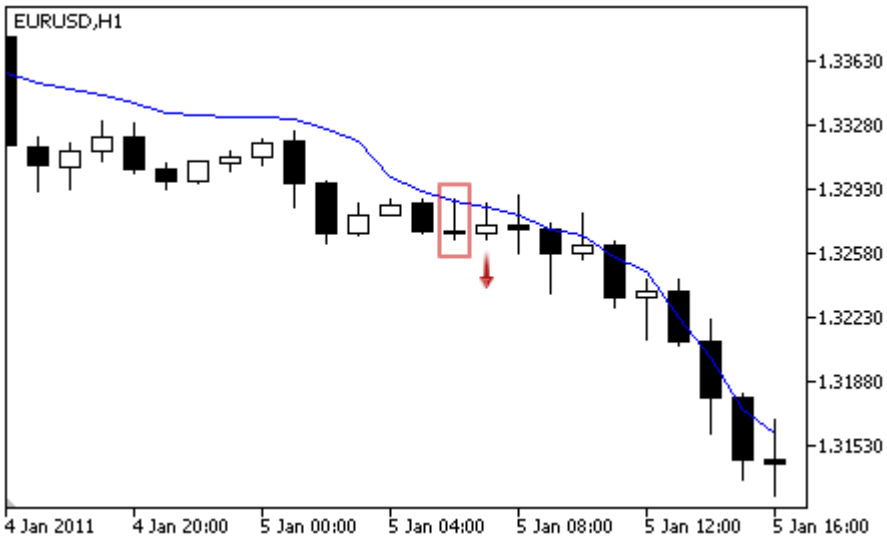
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en hausse (signal fort). 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• L'ombre inférieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont au-dessus de l'indicateur, et le prix le plus bas est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible). 
For selling	• Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en baisse (signal fort).

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • L'ombre supérieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont en-dessous de l'indicateur, et le prix le plus haut est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible). 
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de lissage de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Signaux du Filtre Horaire Intraday

Ce module est basé sur l'assomption que l'efficience des modèles de marché change avec le temps. En utilisant ce module, vous pouvez filtrer les signaux, reçus d'autres modules, par heures et jours de la semaine. Cela permet d'améliorer la qualité des signaux générés grâce à la suppression des périodes de temps les moins favorables. Ce mécanisme de prise de décisions sur la base de signaux des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	Aucun signal.
For selling	Aucun signal.
Aucune objection pour l'achat	L'heure et la date actuelles correspondent aux paramètres spécifiés.
Aucune objection pour la vente	L'heure et la date actuelles correspondent aux paramètres spécifiés.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
GoodHourOfDay	Numéro de l'heure du jour (de 0 à 23) où les signaux seront activés. Si la valeur est -1, les signaux seront activés pour la journée entière.
BadHoursOfDay	Le champ bit. Chaque bit de ce champ correspond à une heure de la journée (bit 0 - heure 0, ..., bit 23 - 23ème heure). Si la valeur d'un bit est égale à 0, les signaux de trading seront activés durant l'heure correspondante. Si la valeur d'un bit est égale à 1, les signaux de trading seront désactivés durant l'heure correspondante. Un nombre spécifié est représenté sous la forme d'un nombre binaire et est utilisé comme un masque de bit. Les heures désactivées ont une plus grande priorité que les heures activées.
GoodDayOfWeek	Numéro du jours de la semaine (de 0 à 6 où 0 est dimanche) où les signaux seront activés. Si la valeur est -1, les signaux seront activés pour tous les jours.
BadDaysOfWeek	Le champ bit. Chaque bit de ce champ correspond à un jour de la semaine (bit 0 - dimanche, ..., bit 6 - samedi). Si la valeur d'un bit est égale à 0, les signaux de trading seront activés durant le jour

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
	correspondant. Si la valeur d'un bit est égale à 1, les signaux de trading seront désactivés durant la journée correspondante. Un nombre spécifié est représenté sous la forme d'un nombre binaire et est utilisé comme un masque de bit. Les jours désactivés ont une plus grande priorité que les jours activés.

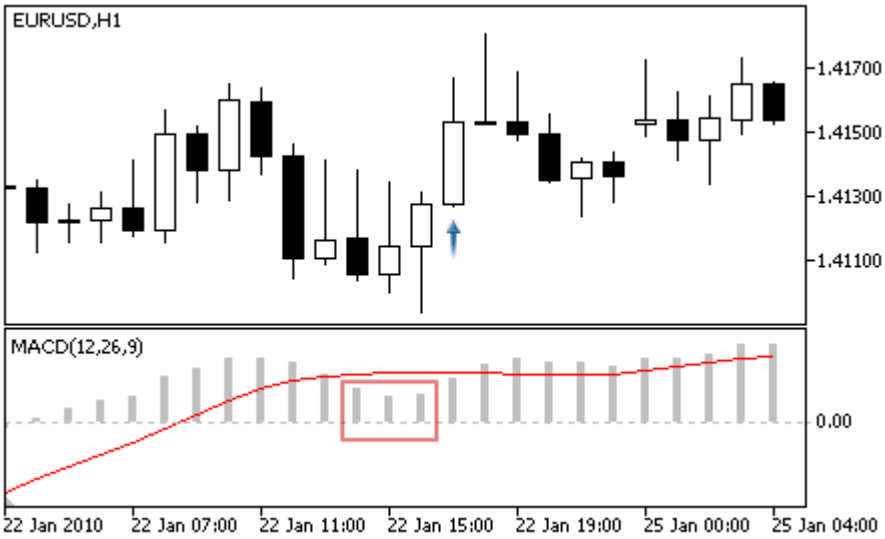
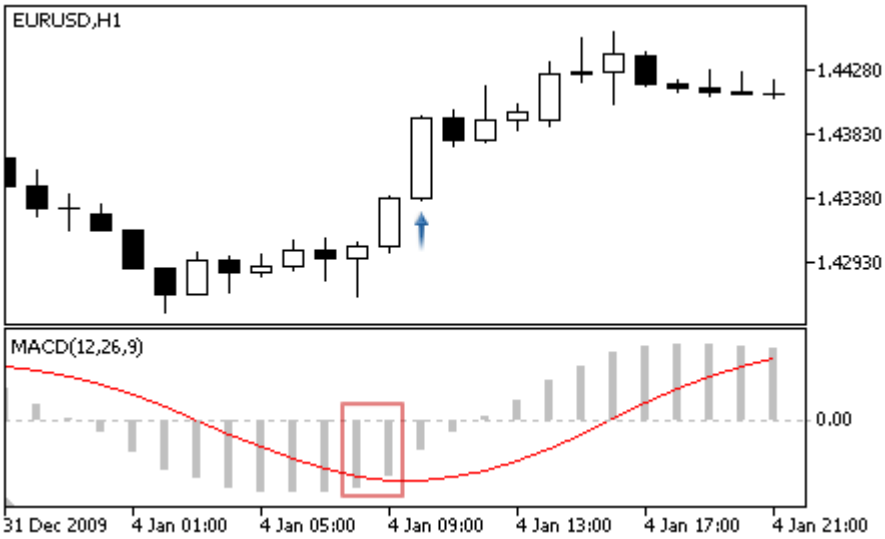
© 2000-2024, MetaQuotes Ltd.

Signaux de l'oscillateur MACD

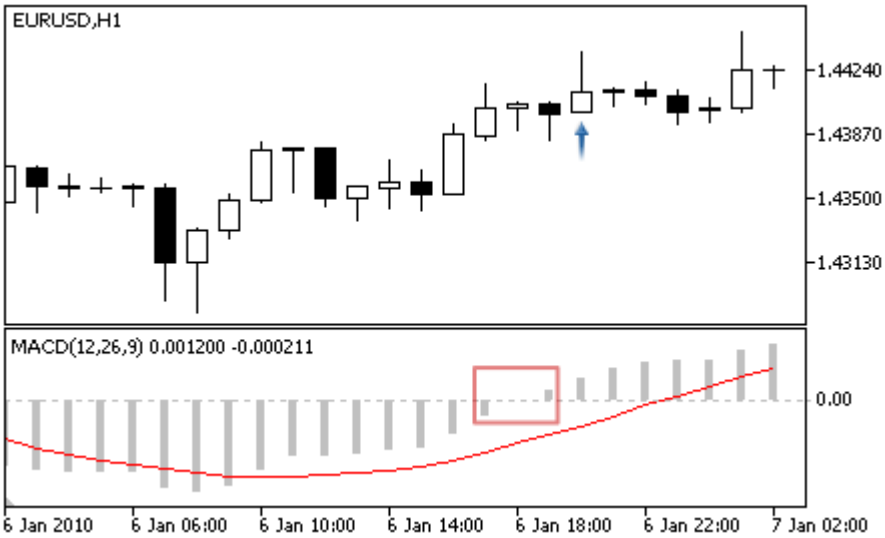
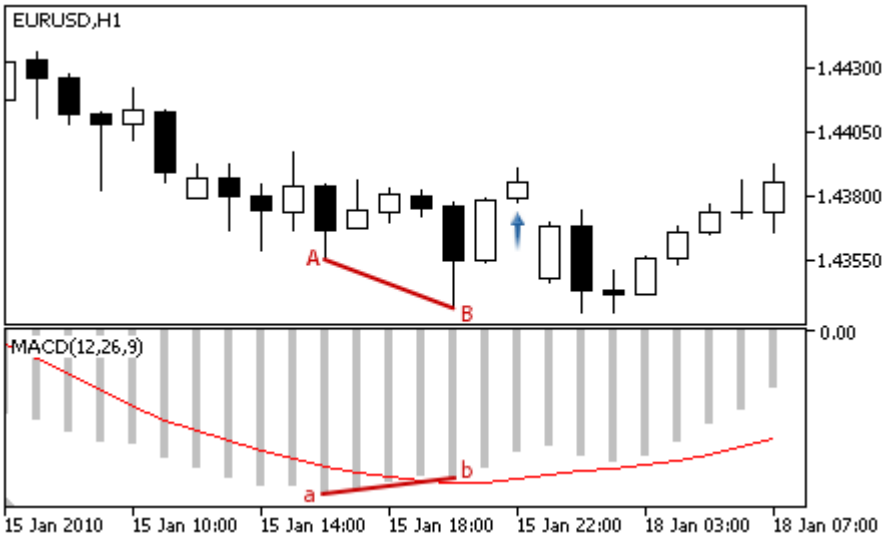
Ce module de signaux est basé sur les modèles de marché de l'oscillateur MACD. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

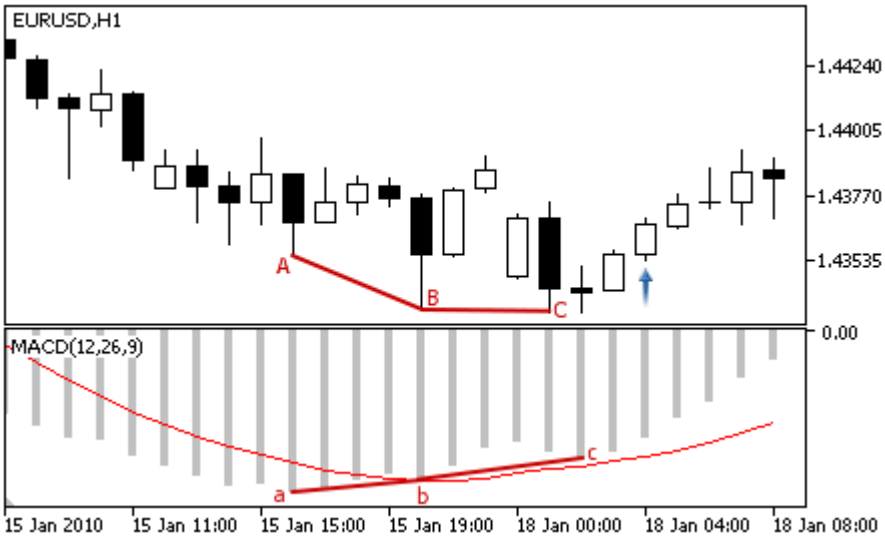
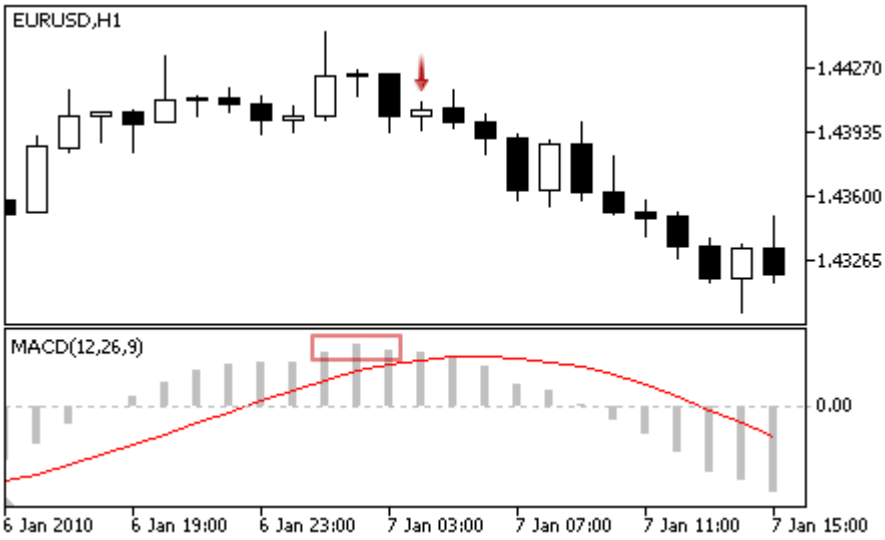
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Renversement** — l'oscillateur s'est retourné à la hausse (l'oscillateur monte sur la barre analysée et baisse sur la barre précédente).  • **Croisement de la ligne de macd et de la ligne de signal** — la ligne de macd est au-dessus de la ligne du signal sur la barre analysée et sous la ligne de signal sur la barre précédente.  • **Croisement avec la ligne zéro** — la ligne de macd est au-dessus de zéro sur la barre analysée et sous la ligne de zéro sur la barre précédente.

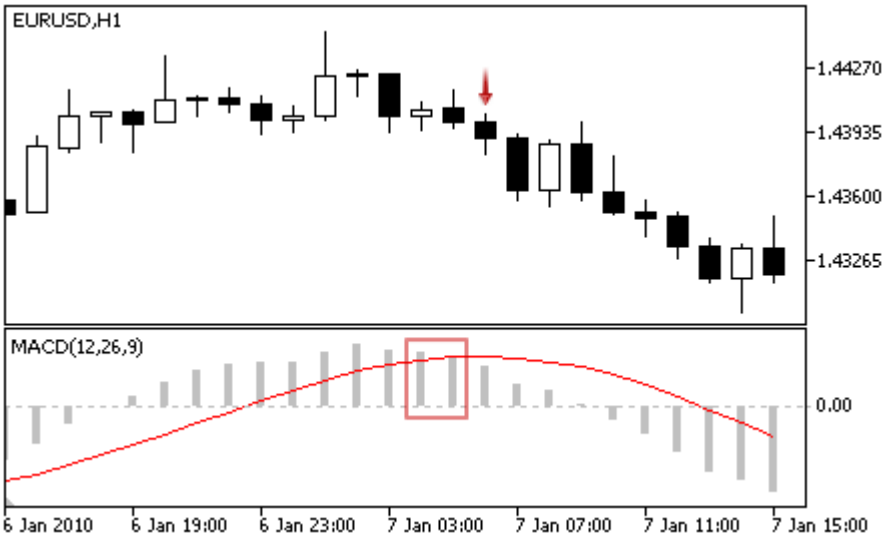
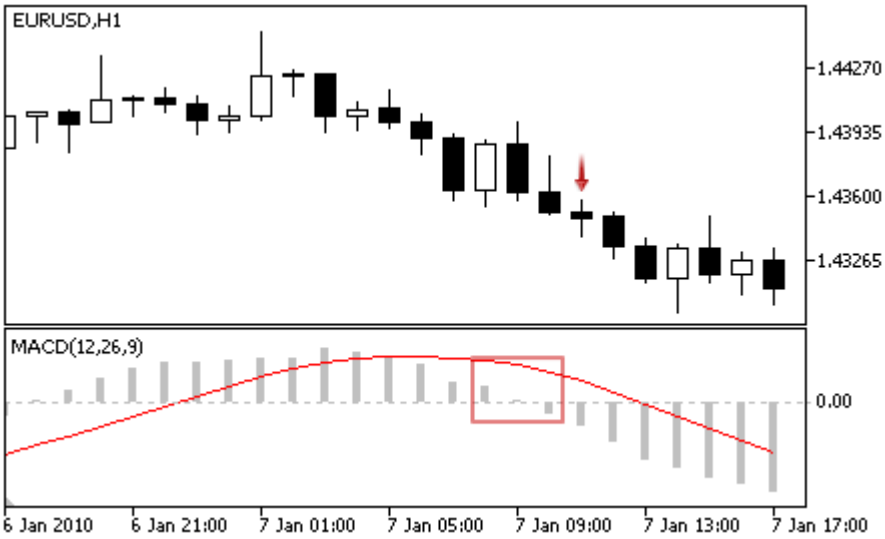
© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent.  • **Double Divergence** — l'oscillateur forme 3 creux consécutifs, chacun d'eux est supérieur au précédent ; et les prix forment également les 3 creux correspondants, et chacun d'eux est plus bas que le précédent.

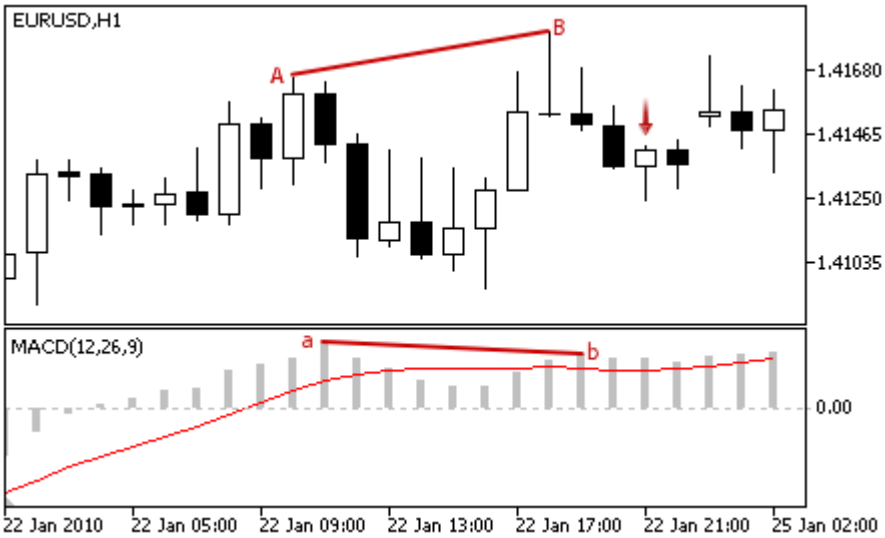
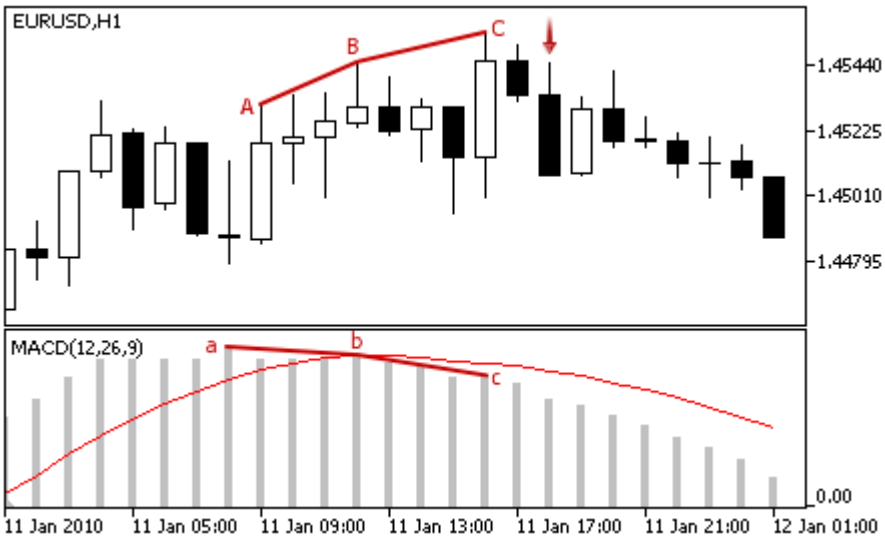
© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	
For selling	• **Renversement** — l'oscillateur se retourne à la baisse (l'oscillateur est en baisse sur la barre analysée et en hausse sur la barre précédente).  • **Croisement de la ligne de macd et de la ligne de signal** — la ligne de macd est en-dessous de la ligne du signal sur la barre analysée et au-dessus de la ligne de signal sur la barre précédente.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Croisement avec la ligne zéro** — la ligne de macd est en-dessous de zéro sur la barre analysée et au-dessus de la ligne de zéro sur la barre précédente.  • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Double Divergence** — l'oscillateur forme 3 pics consécutifs, chacun d'eux est inférieur au précédent ; et les prix forment également les 3 pics correspondants, et chacun d'eux est plus haut que le précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodFast	Période de calcul de la moyenne mobile rapide (fast EMA).
PeriodSlow	Période de calcul de la moyenne mobile lente (slow EMA).
PeriodSignal	Période de lissage.
Applied	Une série de prix utilisée pour le calcul de l'oscillateur.

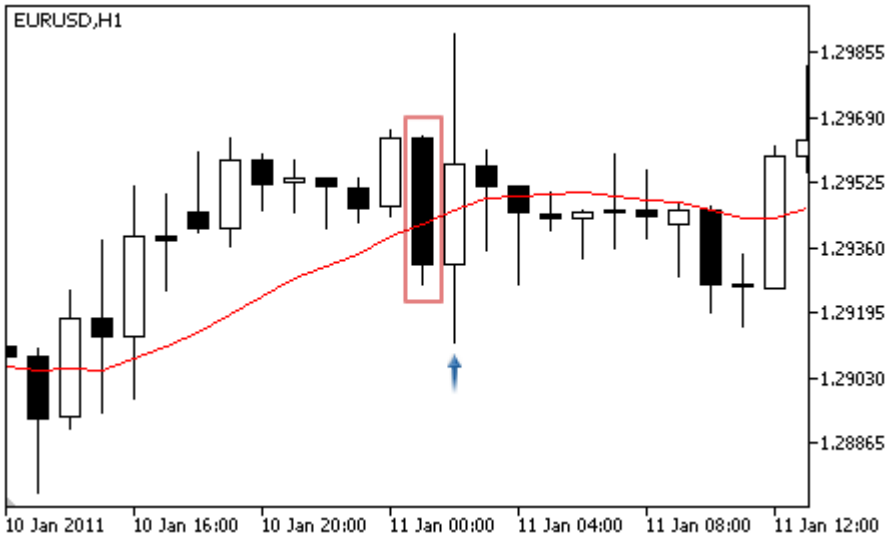
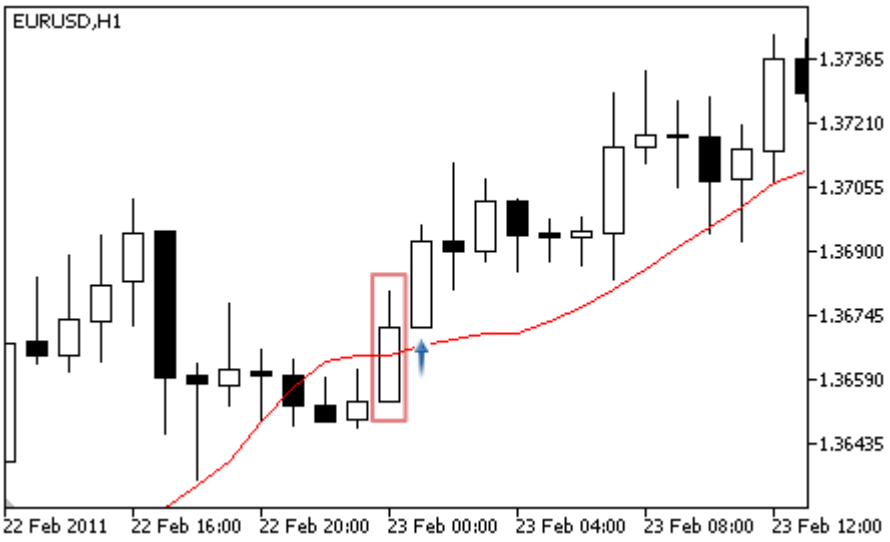
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Moving Average

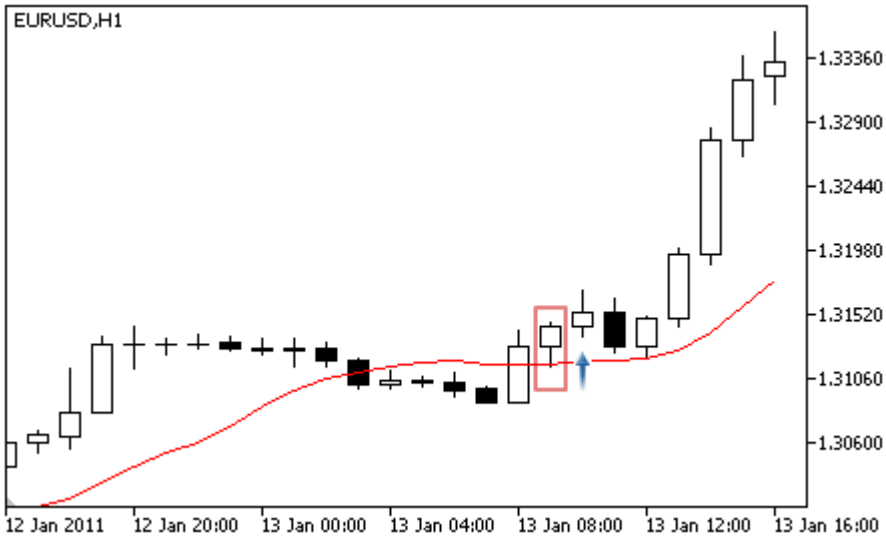
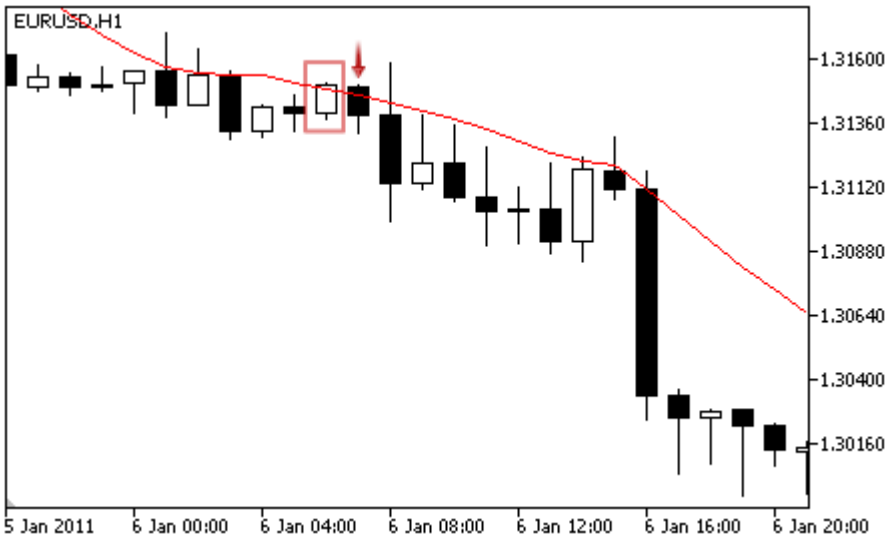
Ce module de signaux est basé sur les modèles de marché de l'indicateurMoyenne Mobile. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

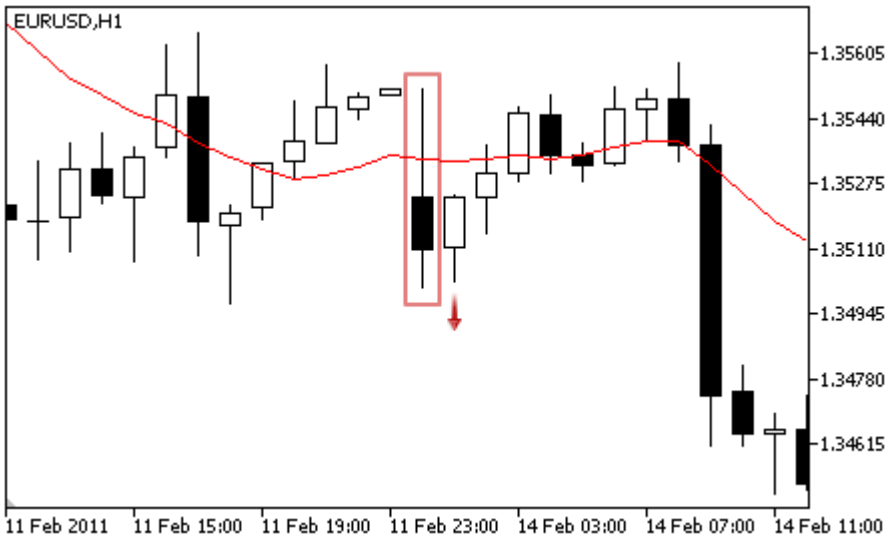
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en hausse (signal fort). 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• L'ombre inférieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont au-dessus de l'indicateur, et le prix le plus bas est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible). 
For selling	• Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en baisse (signal fort).

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • L'ombre supérieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont en-dessous de l'indicateur, et le prix le plus haut est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible). 
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de lissage de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.

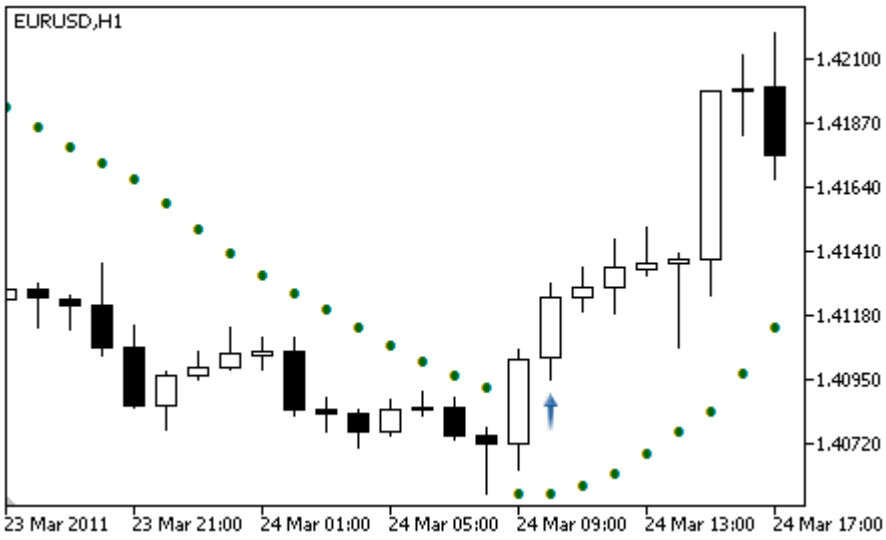
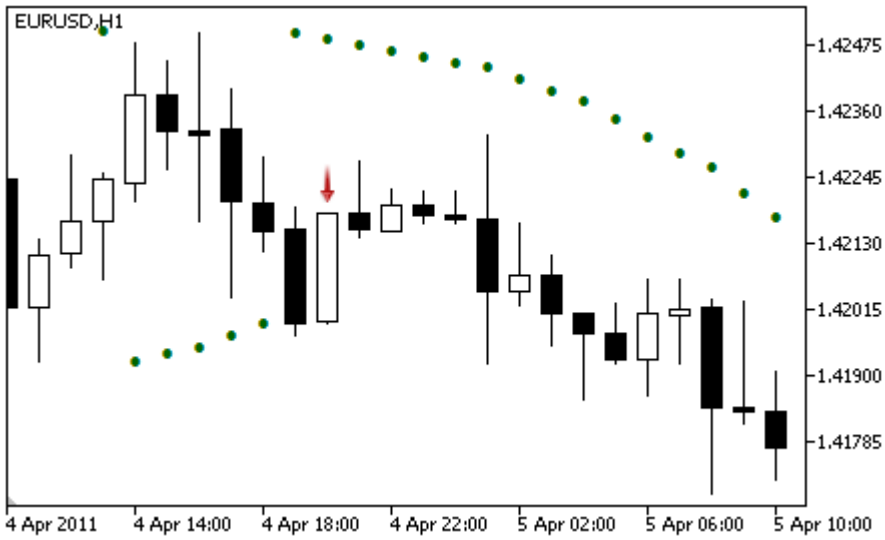
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Parabolic SAR

Ce module de signaux est basé sur les modèles de marché de l'indicateurParabolic SAR. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	**Renversement** — l'indicateur est sous le prix sur la barre analysée et au-dessus du prix sur la barre précédente. 
For selling	**Renversement** — l'indicateur est au-dessus du prix et sous le prix sur la barre précédente. 
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
Step	L'incrément de vitesse de l'indicateur.
Maximum	Taux maximum de la vitesse de convergence de l'indicateur avec le prix.

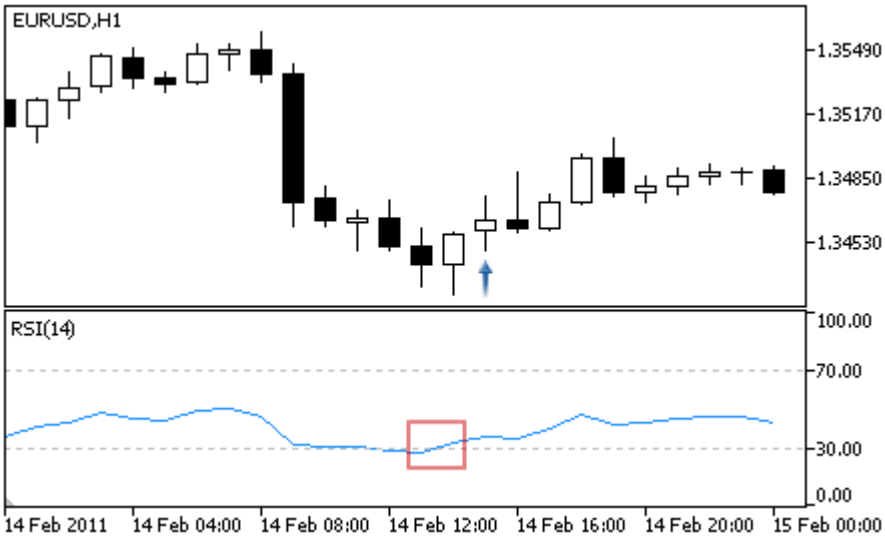
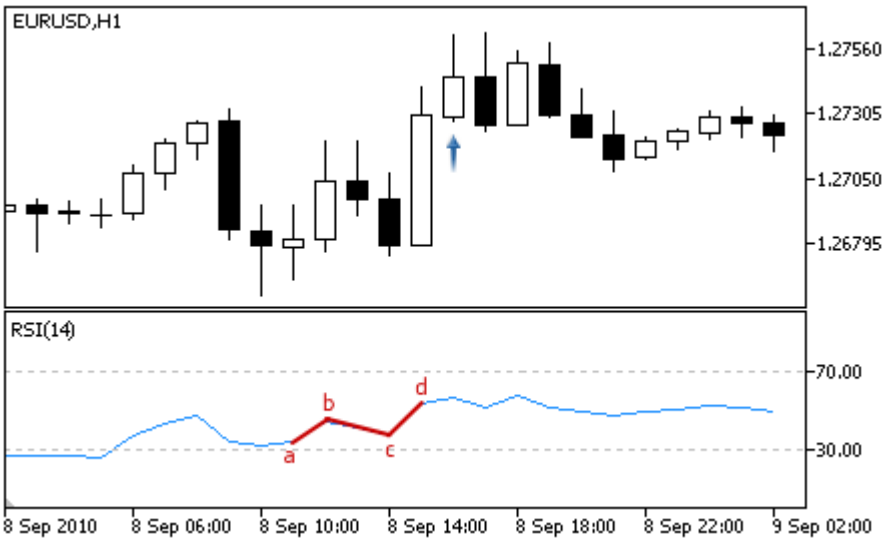
© 2000-2024, MetaQuotes Ltd.

Signaux de l'oscillateur Relative Strength Index

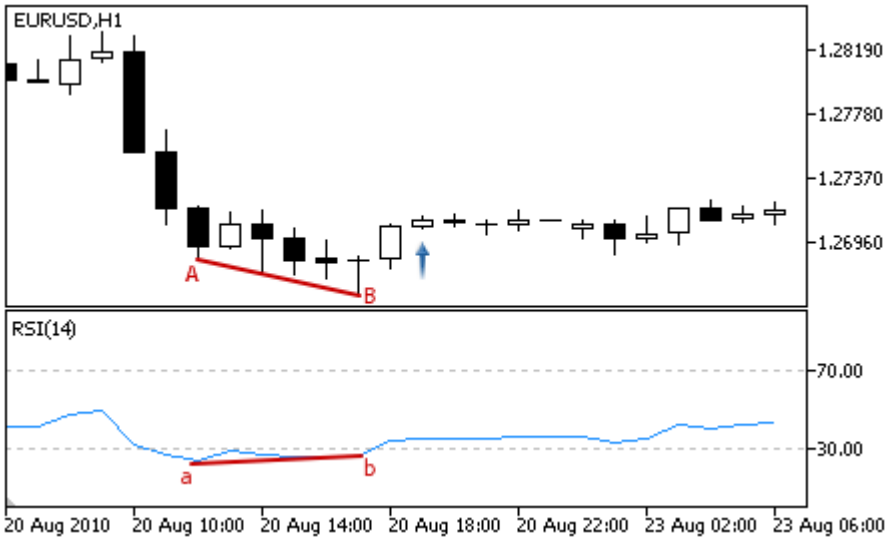
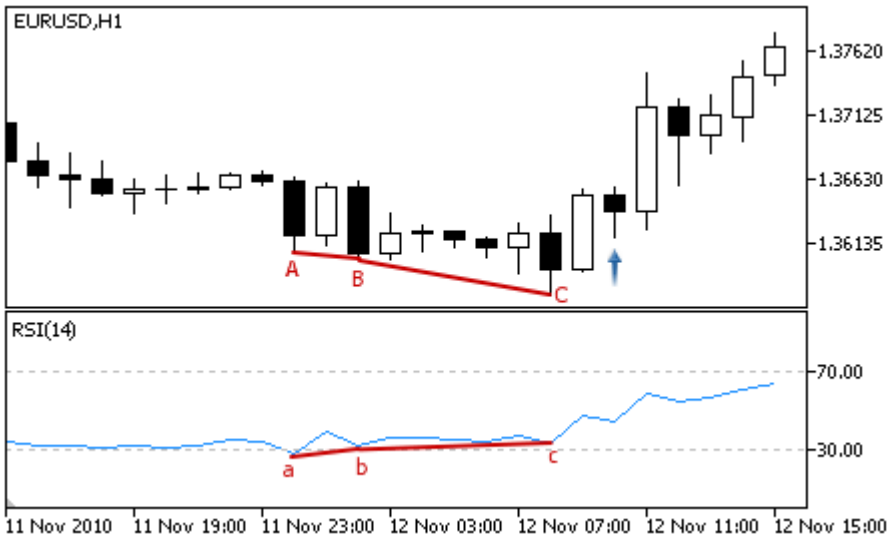
Ce module de signaux est basé sur les modèles de marché de l'oscillateur Relative Strength Index. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

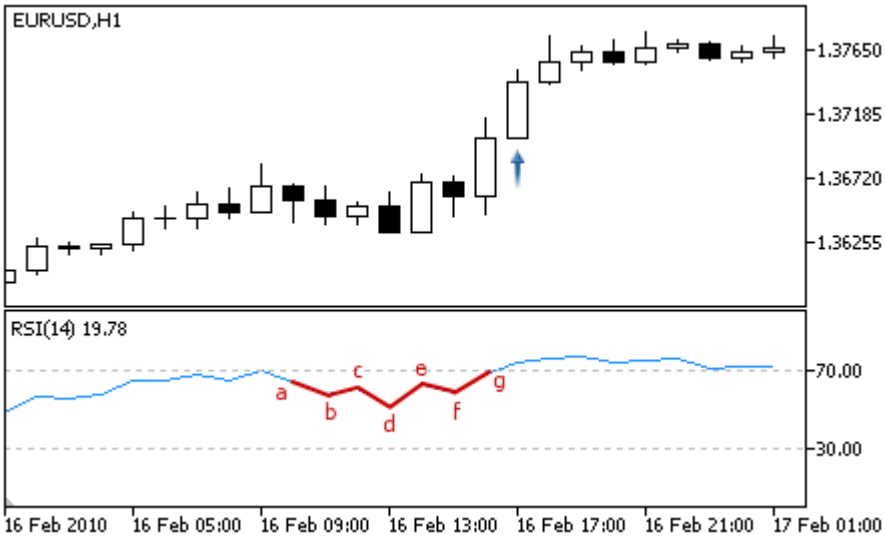
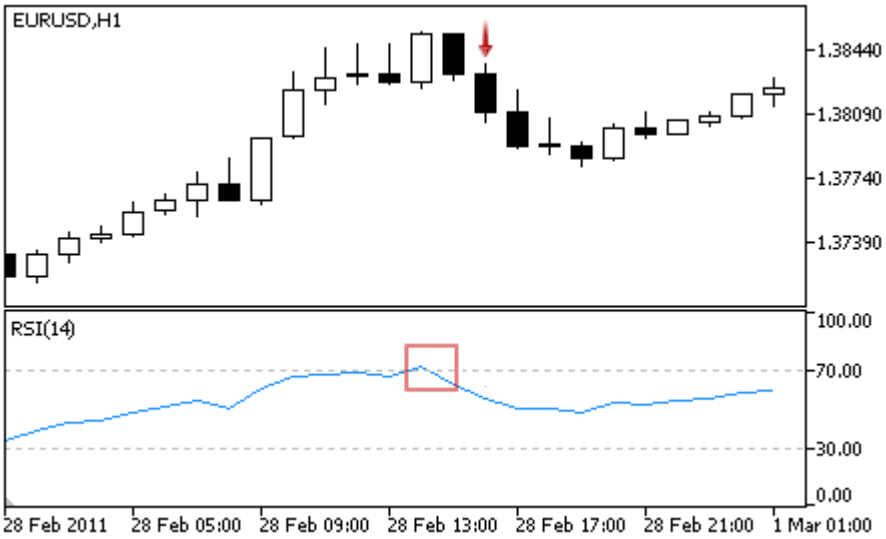
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Retournement dans la zone de sur-vente** — l'oscillateur se retourne à la hausse et sa valeur sur la barre analysée est dans la zone de sur-vente (la valeur par défaut est 30).  • **Failed swing** — l'oscillator monte plus haut que le pic précédent sur la barre analysée. 

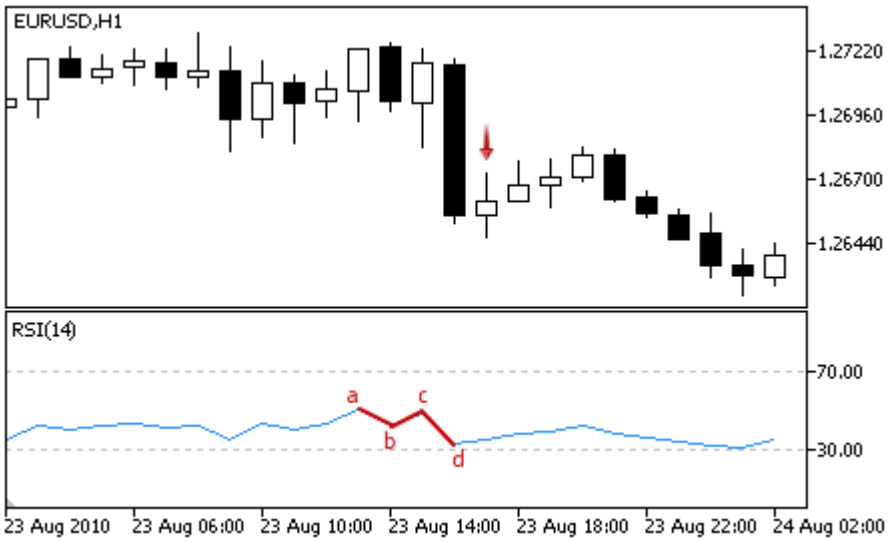
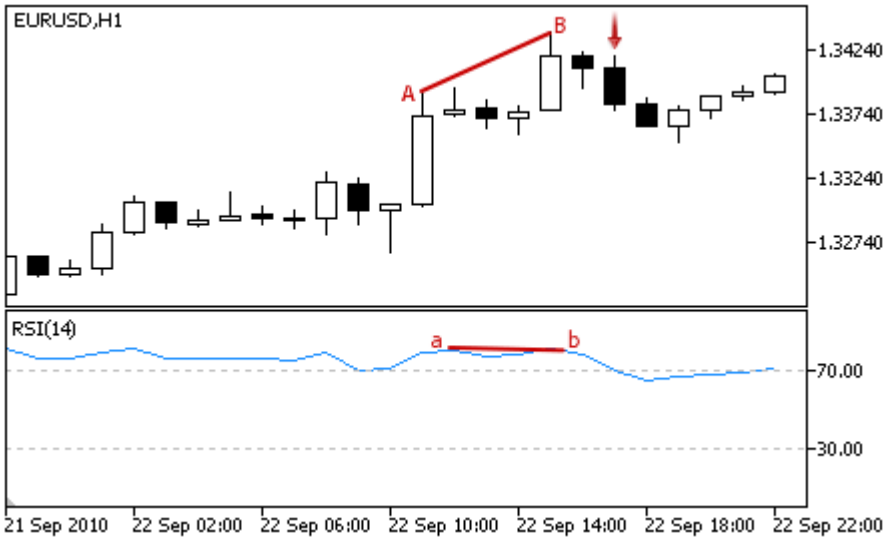
© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• **Divergence** — le premier creux analysé de l'oscillateur est plus bas que le précédent, et le creux correspondant des prix est plus bas que le précédent.  • **Double Divergence** — l'oscillateur forme 3 creux consécutifs, chacun d'eux est supérieur au précédent ; et les prix forment également les 3 creux correspondants, et chacun d'eux est plus bas que le précédent.  • **Tête/Epaules** — l'oscillateur forme trois creux consécutifs et celui du milieu est plus bas que les deux autres.

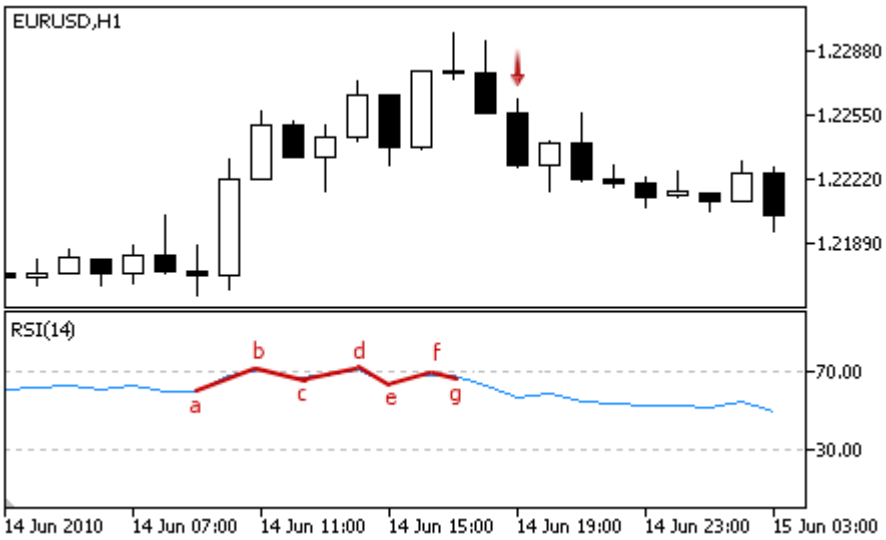
© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	
For selling	• **Renversement en zone de sur-achat** — l'oscillateur se retourne à la baisse et sa valeur sur la barre en cours d'analyse est dans la zone de sur-achat (la valeur par défaut est 70).  • **Failed swing** — l'oscillator baisse plus bas que le creux précédent sur la barre analysée.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent.  • **Double Divergence** — l'oscillateur forme 3 pics consécutifs, chacun d'eux est inférieur au précédent ; et les prix forment également les 3 pics correspondants, et chacun d'eux est plus haut que le précédent.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Tête/Epaules** — l'oscillateur forme 3 pics consécutifs et celui du milieu est plus haut que les 2 autres. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodRSI	Période de calcul de l'oscillateur.
Applied	Une série de prix utilisée pour le calcul de l'oscillateur.

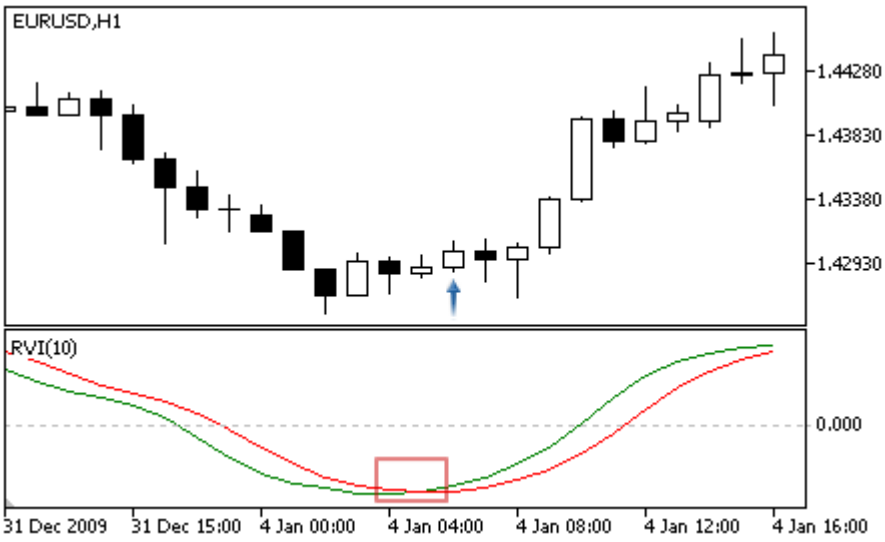
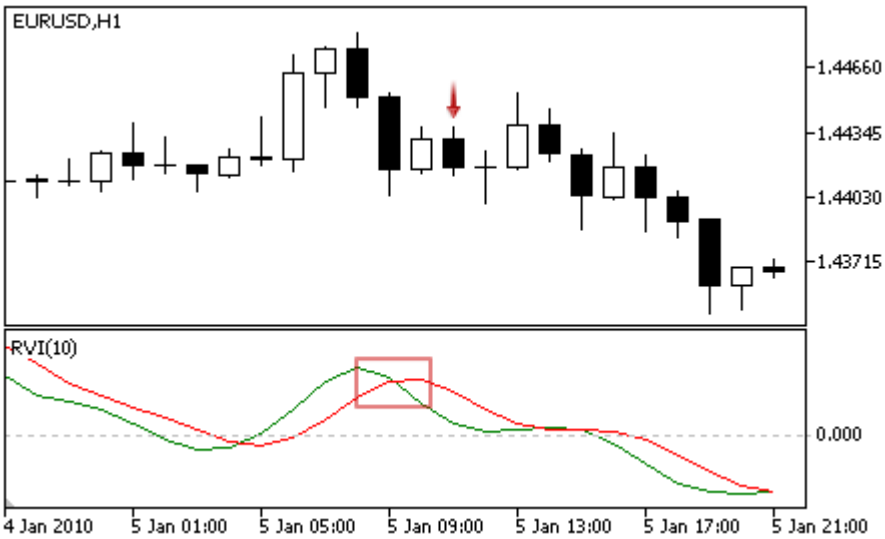
© 2000-2024, MetaQuotes Ltd.

Signaux de l'oscillateur Relative Vigor Index

Ce module de signaux est basé sur les modèles de marché de l'oscillateur Relative Vigor Index. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	**Croisement de la ligne principale et de la ligne de signal** — la ligne principale est au-dessus de la ligne du signal sur la barre analysée et sous la ligne de signal sur la barre précédente. 
For selling	**Croisement de la ligne principale et de la ligne de signal** — la ligne principale est en-dessous de la ligne du signal sur la barre analysée et au-dessus de la ligne de signal sur la barre précédente. 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodRVI	Période de calcul de l'oscillateur.

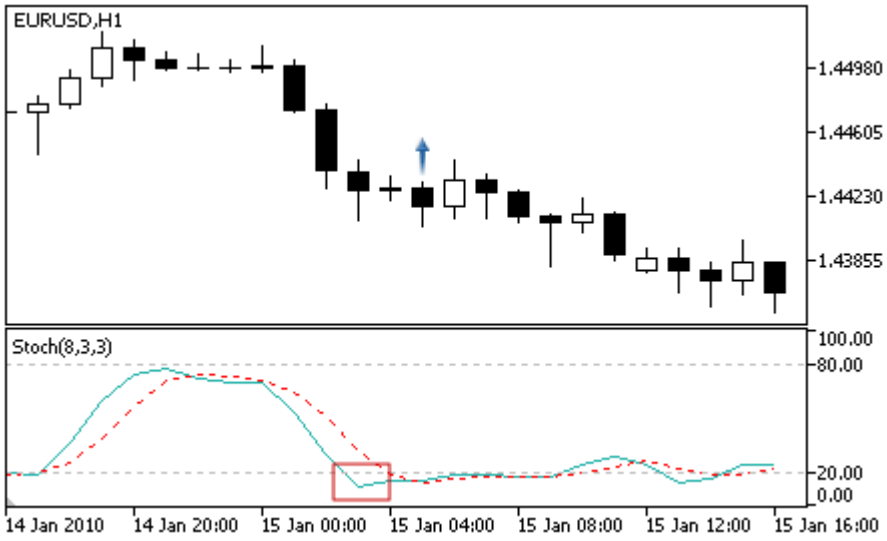
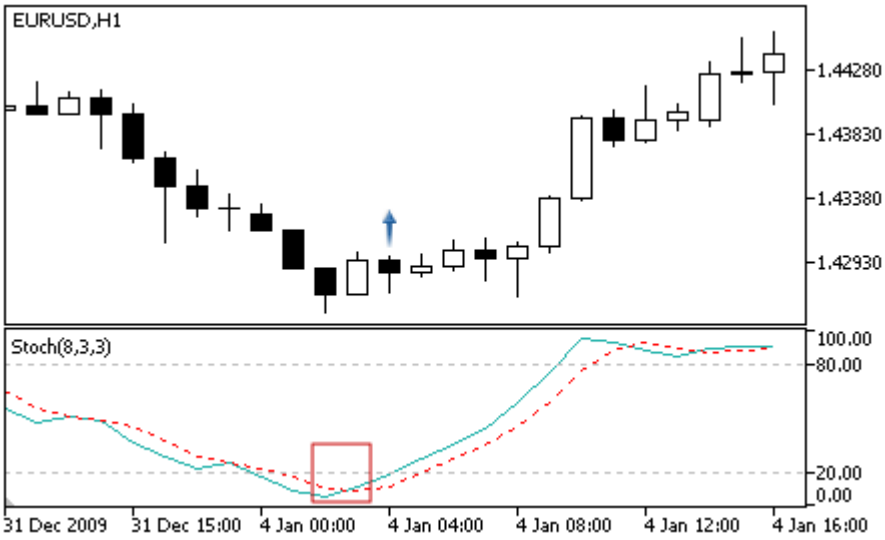
© 2000-2024, MetaQuotes Ltd.

Signaux de l'oscillateur Stochastic

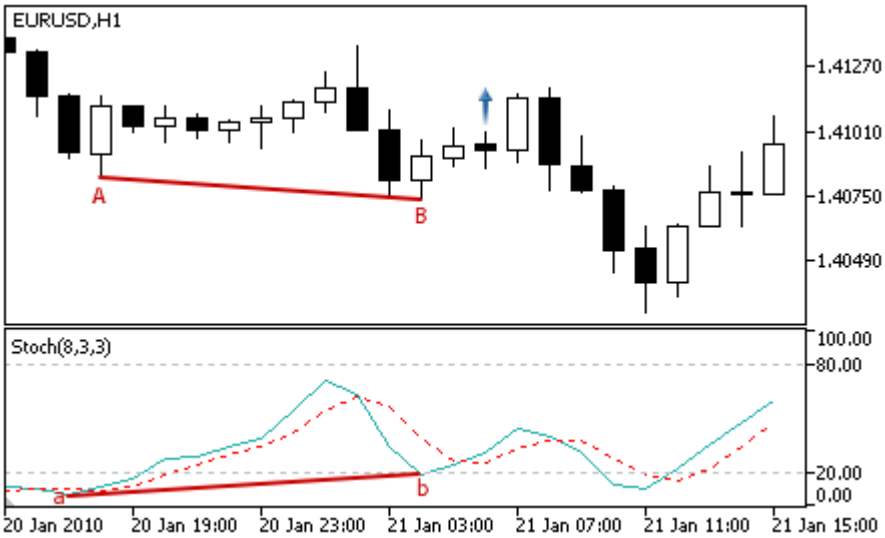
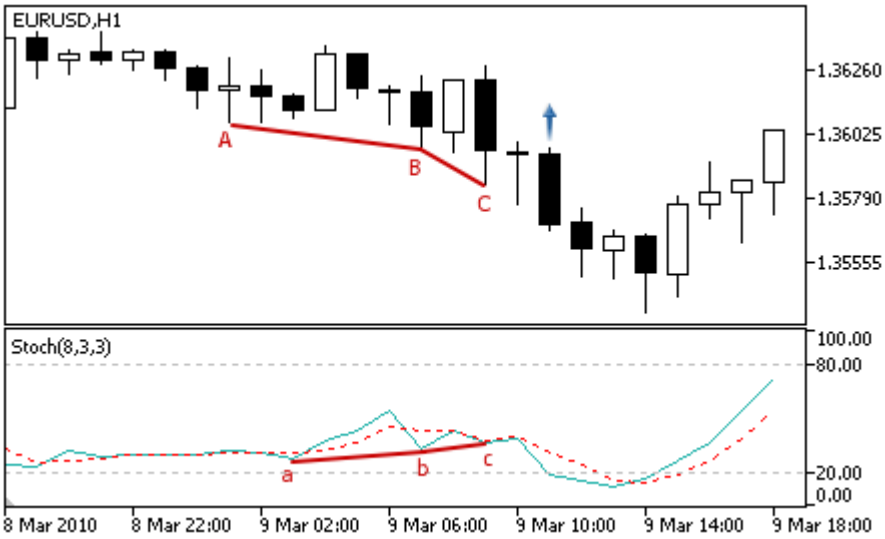
Ce module de signaux est basé sur les modèles de marché de l'oscillateur Stochastique. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

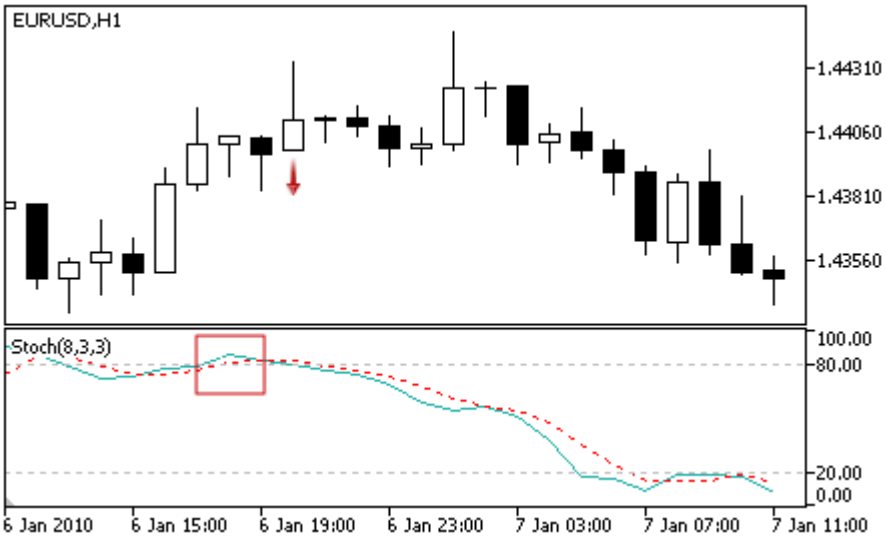
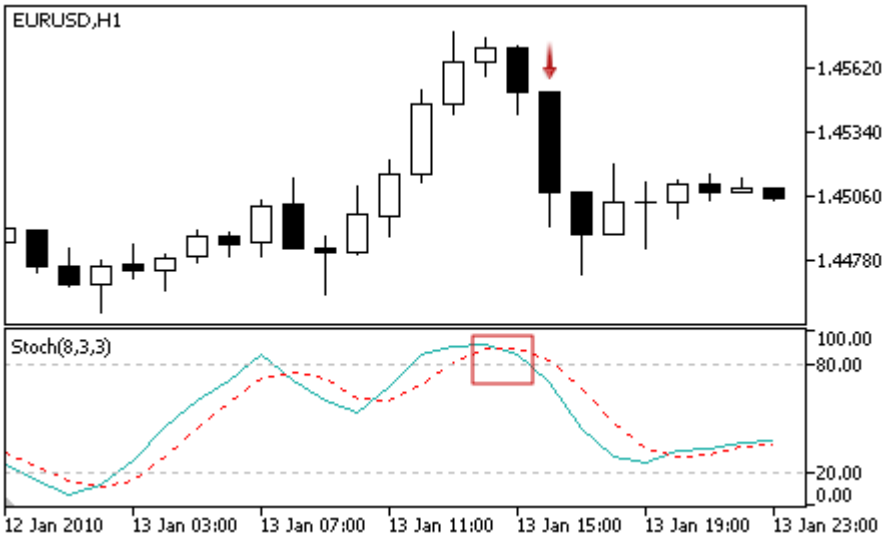
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Renversement** — l'oscillateur s'est retourné à la hausse (l'oscillateur monte sur la barre analysée et baisse sur la barre précédente).  • **Croisement de la ligne principale et de la ligne de signal** — la ligne principale est au-dessus de la ligne du signal sur la barre analysée et sous la ligne de signal sur la barre précédente. 

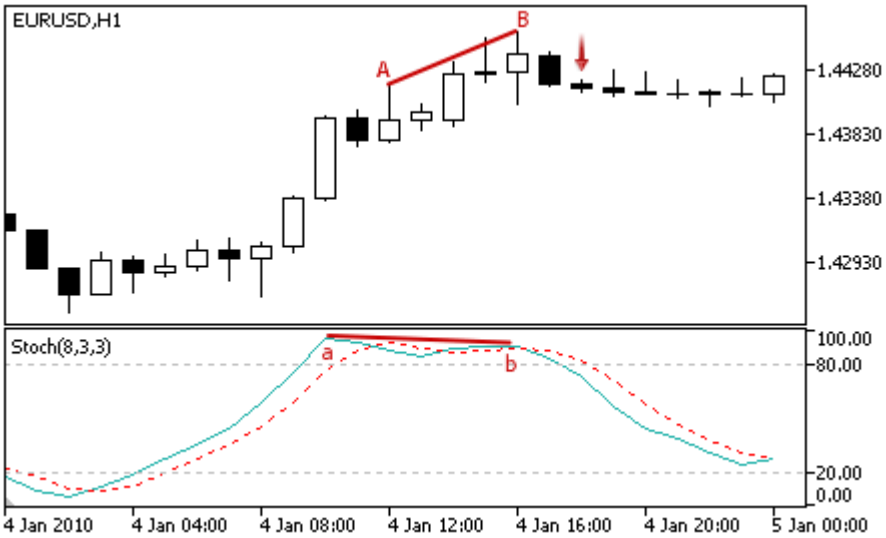
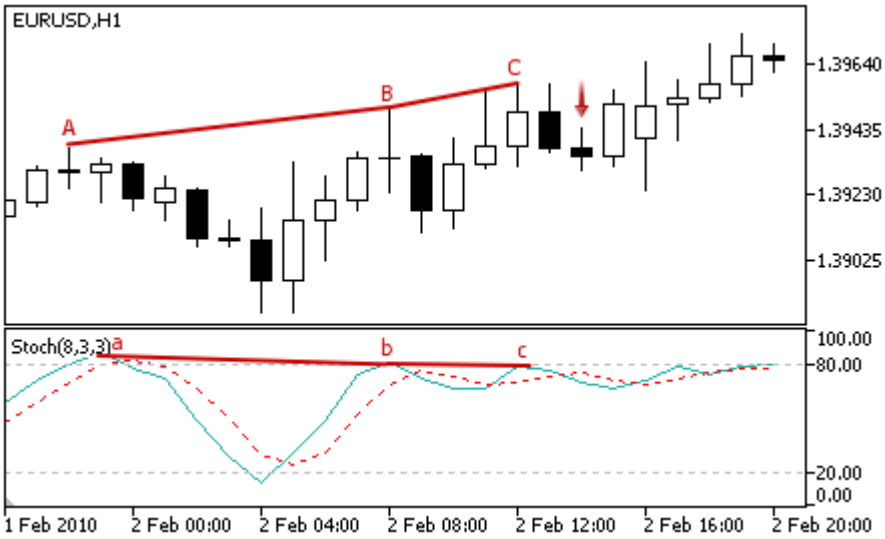
© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent.  • **Double Divergence** — l'oscillateur forme 3 creux consécutifs, chacun d'eux est supérieur au précédent ; et les prix forment également les 3 creux correspondants, et chacun d'eux est plus bas que le précédent. 
For selling	• **Renversement** — l'oscillateur se retourne à la baisse (l'oscillateur est en baisse sur la barre analysée et en hausse sur la barre précédente).

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Croisement de la ligne principale et de la ligne de signal** — la ligne principale est en-dessous de la ligne du signal sur la barre analysée et au-dessus de la ligne de signal sur la barre précédente.  • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Double Divergence** — l'oscillateur forme 3 pics consécutifs, chacun d'eux est inférieur au précédent ; et les prix forment également les 3 pics correspondants, et chacun d'eux est plus haut que le précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodK	Période de calcul de la ligne de signal de l'oscillateur.
PeriodD	Période de calcul de la ligne de signal de l'oscillateur.
PeriodSlow	Période de lissage.
Applied	Une série de prix utilisée pour le calcul de l'oscillateur.

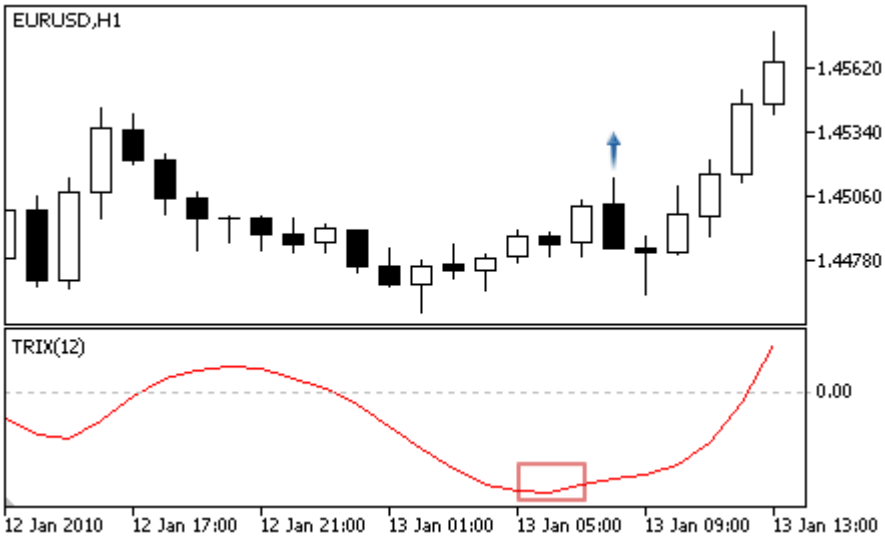
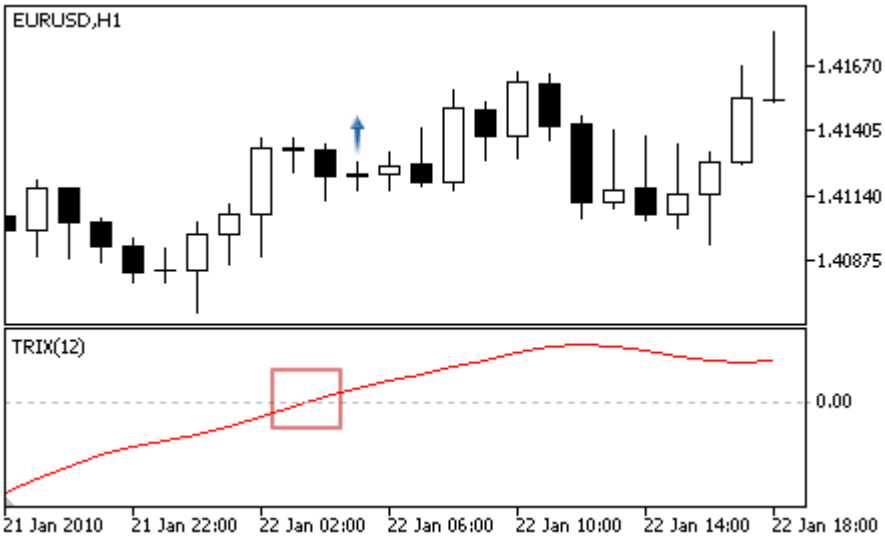
© 2000-2024, MetaQuotes Ltd.

Signaux de l'oscillateur Triple Exponential Average

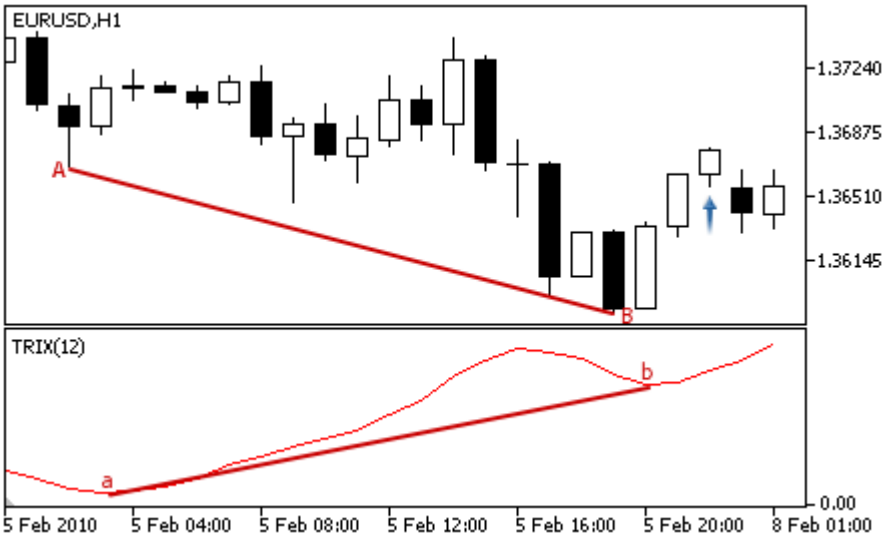
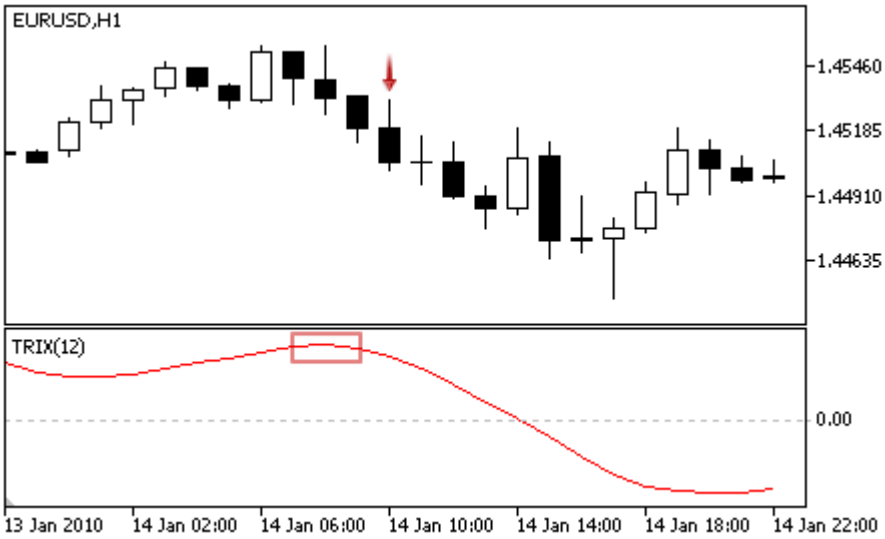
Ce module de signaux est basé sur les modèles de marché de l'indicateurTriple Exponential Average. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

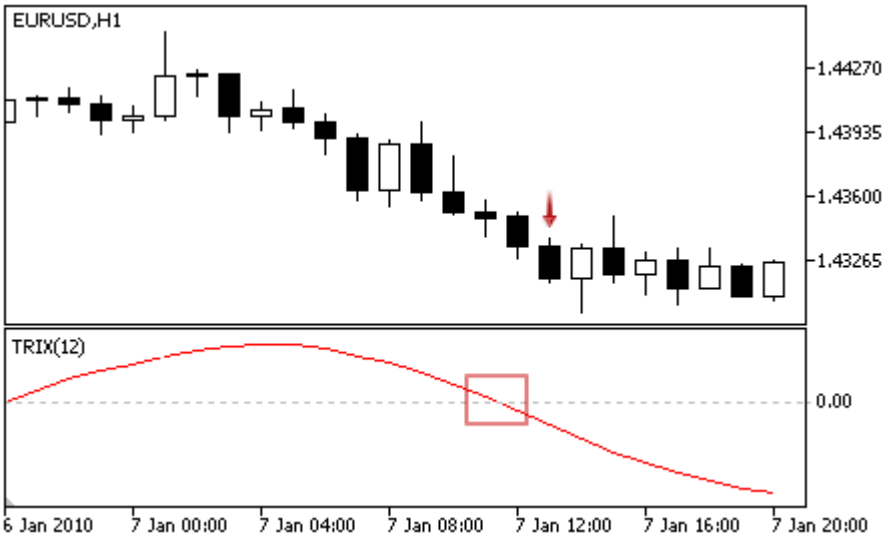
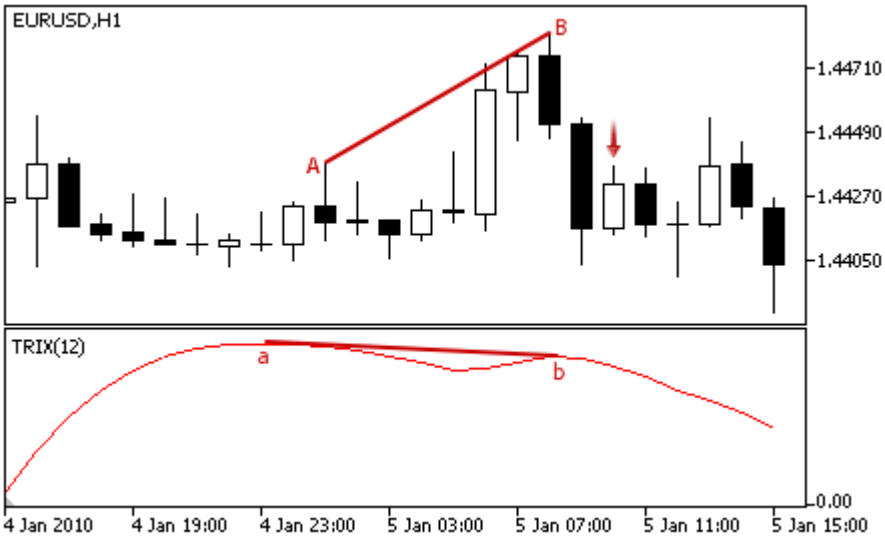
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Renversement** — l'oscillateur s'est retourné à la hausse (l'oscillateur monte sur la barre analysée et baisse sur la barre précédente).  • **Croisement avec la ligne zéro** — la ligne de macd est au-dessus de zéro sur la barre analysée et sous la ligne de zéro sur la barre précédente. 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent. 
For selling	• **Renversement** — l'oscillateur se retourne à la baisse (l'oscillateur est en baisse sur la barre analysée et en hausse sur la barre précédente).  • **Croisement avec la ligne zéro** — la ligne de macd est en-dessous de zéro sur la barre analysée et au-dessus de la ligne de zéro sur la barre précédente.

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodTriX	Période de calcul de l'oscillateur.
Applied	Une série de prix utilisée pour le calcul de l'oscillateur.

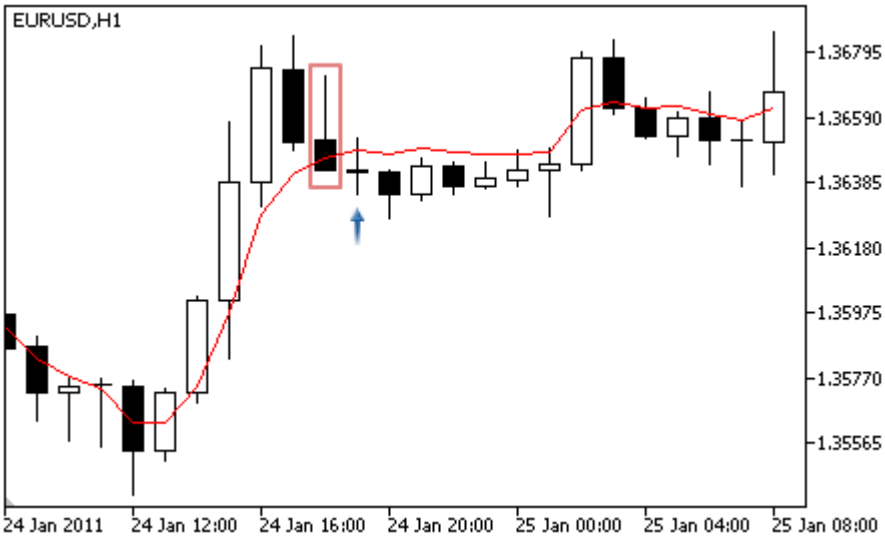
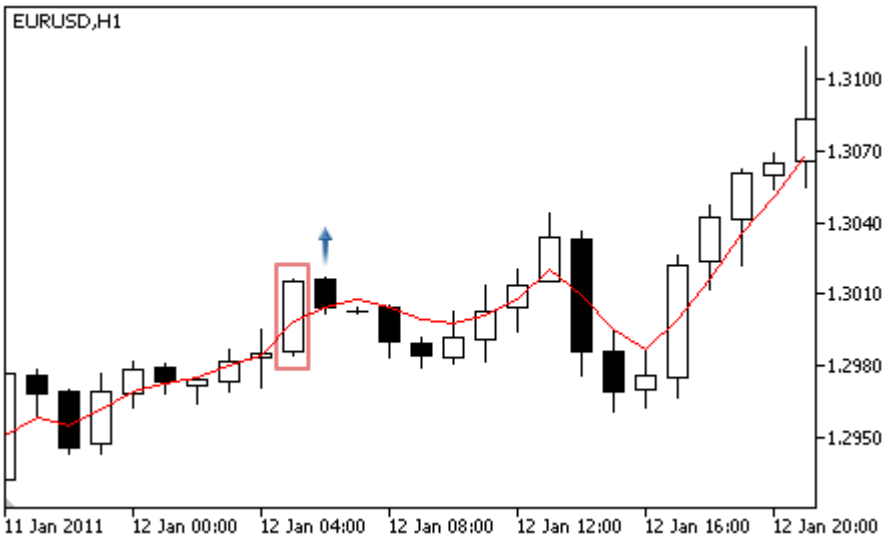
© 2000-2024, MetaQuotes Ltd.

Signaux de l'indicateur Triple Exponential Moving Average

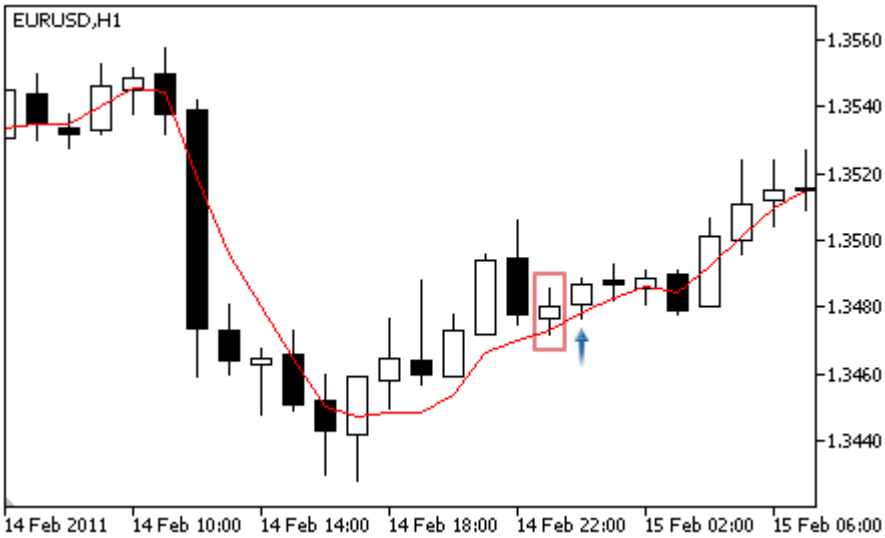
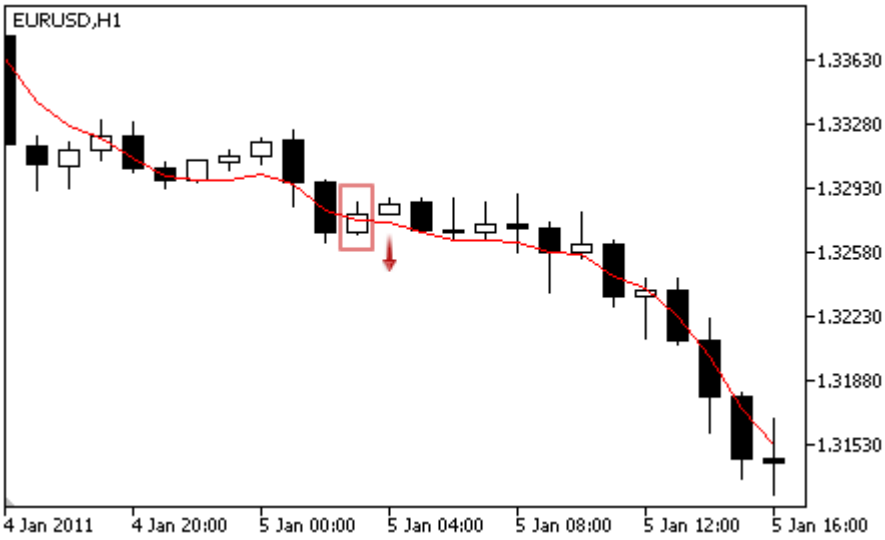
Ce module de signaux est basé sur les modèles de marché de l'indicateurTriple Exponential Moving Average. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

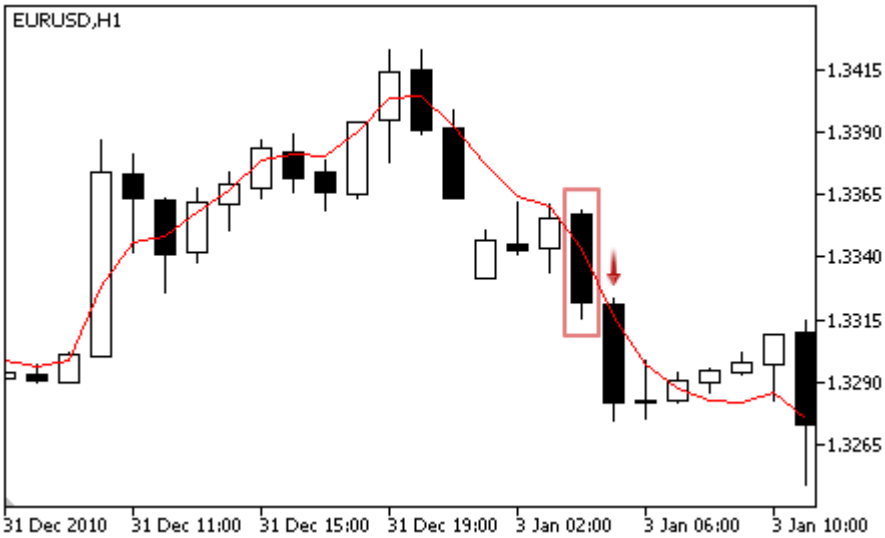
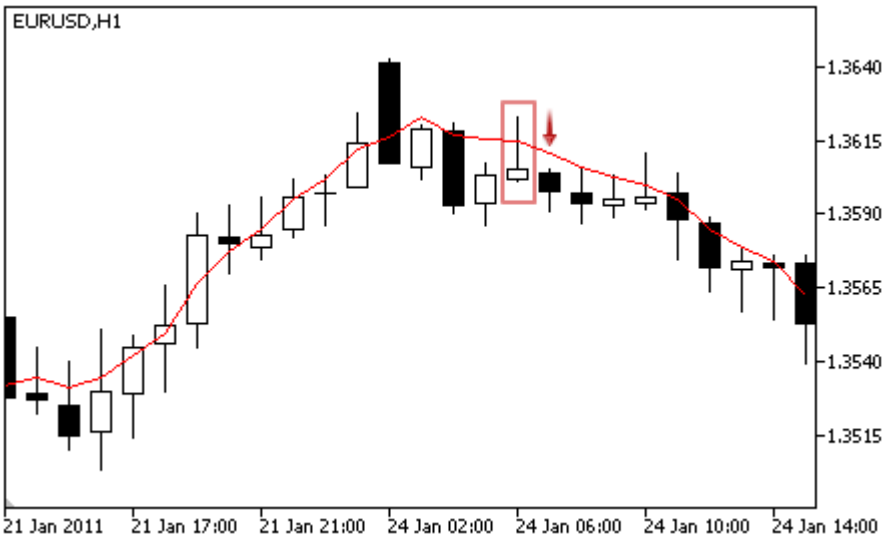
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en hausse (signal fort). 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	• L'ombre inférieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont au-dessus de l'indicateur, et le prix le plus bas est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible). 
For selling	• Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en baisse (signal fort).

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
	 • L'ombre supérieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont en-dessous de l'indicateur, et le prix le plus haut est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible). 
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

© 2000-2024, MetaQuotes Ltd.

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de lissage de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.

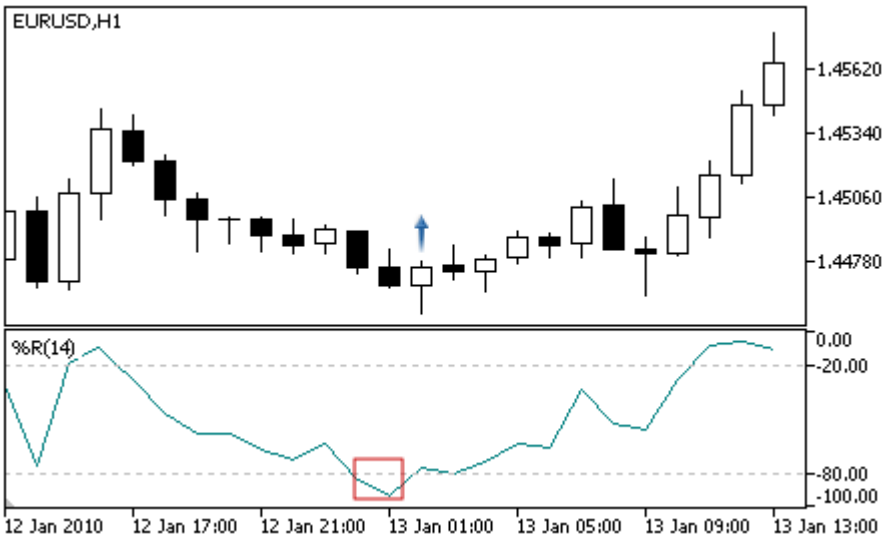
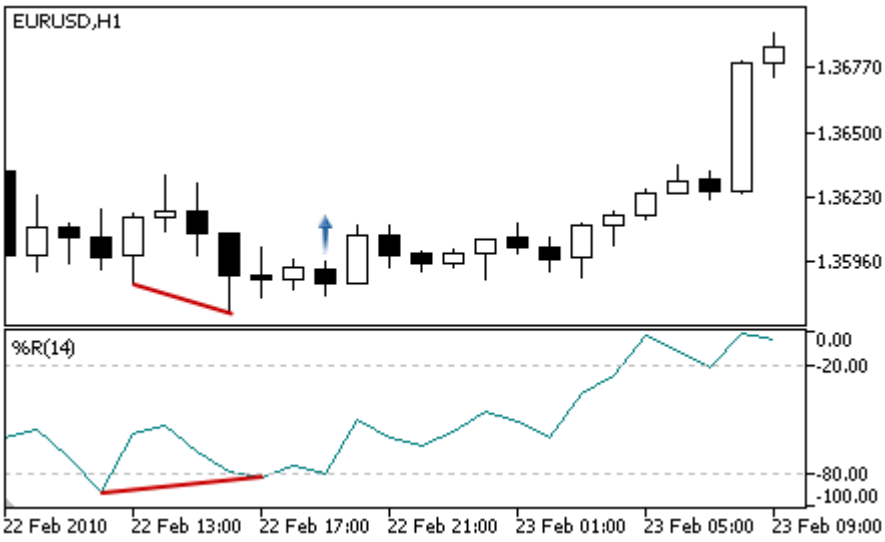
© 2000-2024, MetaQuotes Ltd.

Signaux de l'oscillatuer Williams Percent Range

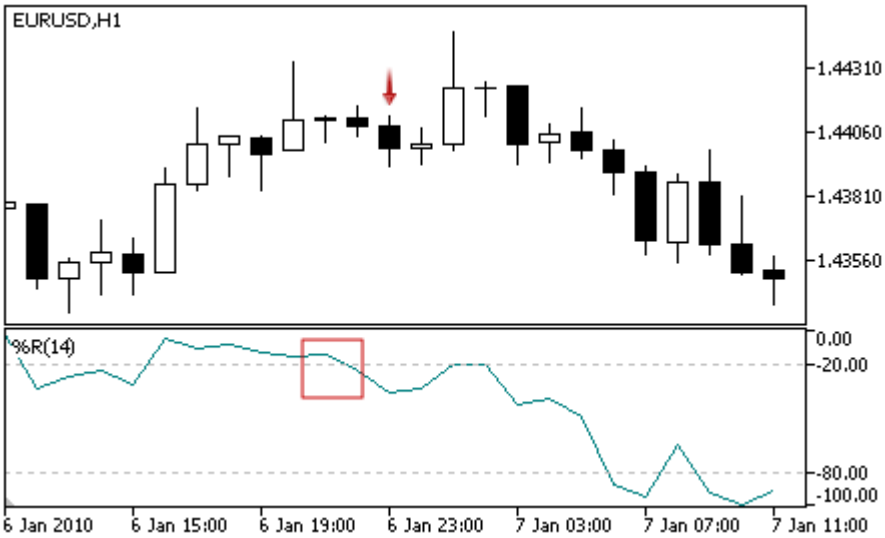
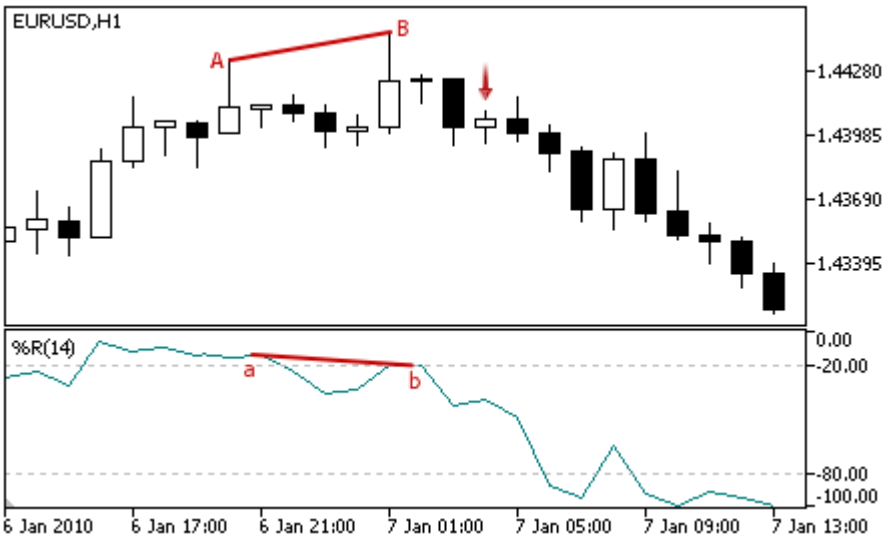
Ce module de signaux est basé sur les modèles de marché de l'oscillateur Williams Percent Range. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Retournement dans la zone de sur-vente** — l'oscillateur se retourne à la hausse et sa valeur sur la barre analysée est dans la zone de sur-vente (la valeur par défaut est -20).  • **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent. 

© 2000-2024, MetaQuotes Ltd.

Type de Signal	Description des Conditions
For selling	• **Renversement en zone de sur-achat** — l'oscillateur se retourne à la baisse et sa valeur sur la barre en cours d'analyse est dans la zone de sur-achat (la valeur par défaut est -80).  • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

© 2000-2024, MetaQuotes Ltd.

Noter que l'oscillateur Williams Percent Range a une échelle inversée. Sa valeur maximum est -100, et la minimum est 0.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodWPR	Période de calcul de l'oscillateur.

© 2000-2024, MetaQuotes Ltd.

Classes de Trailing Stop (Stops Suiveurs)

Cette section contient les détails techniques d'utilisation des classes de stops suiveurs et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création (et du test) de vos stratégies de trading.

La Bibliothèque Standard MQL5 (en terme de stratégies de trading) est située dans le répertoire de travail du terminal dans le répertoire Include\Expert\Trailing.

Classe	Description
CTrailingFixedPips	Cette classe implémente un algorithme de stop suiveur, basé sur des points fixes
CTrailingMA	Cette classe implémente un algorithme de stop suiveur, basé sur les valeurs d'une moyenne mobile
CTrailingNone	Classe de type stub, elle n'utilise aucun algorithme de Stop Suiveur
CTrailingPSAR	Cette classe implémente un algorithme de stop suiveur, basé sur les valeurs de l'indicateur Parabolic SAR

© 2000-2024, MetaQuotes Ltd.

CTrailingFixedPips

La classe CTrailingFixedPips implémente un algorithme de stop suiveur, basé sur un suivi avec des points fixes.

Si la position a un Stop Loss, vérifie la distance minimale autorisée du Stop Loss au prix courant. Si sa valeur est inférieure au prix du Stop Loss, cela suggère de définir un nouveau prix pour le Stop Loss. Si la position a un Take Profit, suggère de définir un nouveau prix pour le Take Profit.

Si l'Expert Advisor a été initialisé avec le flag every_tick=faux, la class n'effectuera toutes les opérations (trading, suivi, etc.) que sur la nouvelle barre. Dans ce cas, le niveau du Take Profit peut être utilisé. Cela vous permettra de clôturer la position ouverte au prix du Take Profit avant que la nouvelle barre ne soit complétée.

Description

CTrailingFixedPips implémente l'algorithme du Stop Suiveur basé sur une taille fixe par rapport aux positions.

Déclaration

```
class CTrailingFixedPips: public CExpertTrailing
```

Titre

```
#include <Expert\Trailing\CTrailingFixedPips.mqh>
```

Hiérarchie d'héritage

CObject
- CExpertBase
  - CExpertTrailing
    - CTrailingFixedPips

Méthodes de Classe

Initialisation	
StopLevel	Définit la valeur du niveau du Stop Loss
ProfitLevel	Définit la valeur du niveau du Take Profit
virtual ValidationSettings	Vérifie les paramètres
Méthodes de Vérifications des Suivis	
virtual CheckTrailingStopLong	Vérifie les conditions du Stop Suiveur d'une position longue
virtual CheckTrailingStopShort	Vérifie les conditions du Stop Suiveur d'une position courte

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2024, MetaQuotes Ltd.

StopLevel

Définit la valeur du niveau du Stop Loss (en points).

```
void  StopLevel(
   int  stop_level      // Stop Loss
   )
```

Paramètres

stop_loss

[in] La Nouvelle valeur du niveau du Stop Loss (selon la convention des points à 2/4 décimales).

Note

Si le niveau du Stop Loss est égal à 0, le Stop Suiveur n'est pas utilisé.

© 2000-2024, MetaQuotes Ltd.

ProfitLevel

Définit la valeur du niveau du Take Profit (en points).

```
void  ProfitLevel(
   int  profit_level      // Take profit
   )
```

Paramètres

profit_level

[in] Nouvelle valeur du niveau du Take Profit (selon la convention des points à 2/4 décimales).

Note

Si le niveau du Take Profit est égal à 0, le Stop Suiveur n'est pas utilisé.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La fonction vérigie les niveaux du Take Profit et du Stop Loss. Les valeurs correctes sont 0 et les valeurs supérieures au stop minimal des ordres stops du symbole.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Vérifie les conditions du Stop Suiveur d'une position longue.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // pointeur sur un objet CPositionInfo
   double&         sl,            // Prix du Stop Loss
   double&         tp             // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Si le niveau du Stop Loss est égal à 0, le Stop Suiveur n'est pas utilisé. Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est considérée comme le prix de base.

Si le prix Bid (prix de l'offre) est supérieur au prix de base+stop loss, cela suggère de définir un nouveau prix de Stop Loss. Dans ce cas, si la position a déjà un Take Profit, cela suggère de définir un nouveau Take Profit égal au niveau prix Bid +Take Profit.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Vérifie les conditions du Stop Suiveur d'une position courte.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // pointeur sur un objet CPositionInfo
   double&         sl,            // Prix du Stop Loss
   double&         tp             // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Si le niveau du Stop Loss est égal à 0, le Stop Suiveur n'est pas utilisé. Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est considérée comme le prix de base.

If the current Ask price is lower than base price-stop loss level, it suggests to set new Prix du Stop Loss Dans ce cas, si la position a déjà un Take Profit, cela suggère de définir un nouveau Take Profit égal au niveau prix Ask - Take Profit.

© 2000-2024, MetaQuotes Ltd.

CTrailingMA

La classe CTrailingMA implémente un algorithme de stop suiveur, basé sur les valeurs d'une moyenne mobile.

Description

La classe CTrailingMA implémente un algorithme de stop suiveur, basé sur les valeurs d'une moyenne mobile

Déclaration

```
class CTrailingMA: public CExpertTrailing
```

Titre

```
#include <Expert\Trailing\TrailingMA.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertTrailing
      CTrailingMA

Méthodes de Classe

Initialisation	
Period	Définit la période de la moyenne mobile.
Shift	Définit le décalage de la moyenne mobile.
Method	Définit la méthode de lissage de la moyenne mobile.
Applied	Définit le prix à appliquer à la moyenne mobile.
virtual InitIndicators	Initialise les indicateurs et les séries de données.
virtual ValidationSettings	Vérifie les paramètres
Méthodes de Vérifications des Suivis	
virtual CheckTrailingStopLong	Vérifie les conditions du Stop Suiveur d'une position longue
virtual CheckTrailingStopShort	Vérifie les conditions du Stop Suiveur d'une position courte

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2024, MetaQuotes Ltd.

Period

Définit la période de la moyenne mobile.

```
void  Period(
   int  period      // Période de lissage
   )
```

Paramètres

period

[in] Période de la moyenne mobile.

© 2000-2024, MetaQuotes Ltd.

Shift

Définit le décalage de la moyenne mobile.

```
void  Shift(
   int  shift      // Décalage
   )
```

Paramètres

shift

[in] Décalage de la moyenne mobile.

© 2000-2024, MetaQuotes Ltd.

Method

Définit la méthode de lissage de la moyenne mobile.

```
void  Method(
   ENUM_MA_METHOD  method      // Méthode de lissage
   )
```

Paramètres

method

[in] Méthode de lissage de la moyenne mobile.

© 2000-2024, MetaQuotes Ltd.

Applied

Définit le prix à appliquer à la moyenne mobile.

```
void  Applied(
   ENUM_APPLIED_PRICE  applied      // Prix appliqué
   )
```

Paramètres

applied

[in] Prix appliqué à la moyenne mobile.

© 2000-2024, MetaQuotes Ltd.

InitIndicators

Initialise les indicateurs et les séries de données.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // Pointeur vers une collection de CIndicators
   )
```

Paramètres

indicators

[in] Pointeur vers une collection d'indicateurs et de séries de données (membre de la classe CExpert).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La fonction vérifie la période de la moyenne mobile, les valeurs correctes sont positives.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Vérifie les conditions du Stop Suiveur d'une position longue.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // pointeur sur un objet CPositionInfo
   double&         sl,            // Prix du Stop Loss
   double&         tp             // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Le prix maximum du Stop Loss est d'abord calculé, le plus près du prix courant en utilisant les valeurs de la moyenne mobile de la barre précédente (barre complète).

Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est utilisé.

Si le prix calculé du Stop Loss est supérieur au prix de base et inférieur au prix maximum du Stop Loss autorisé, cela suggère de définir un nouveau prix de Stop Loss.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Vérifie les conditions du Stop Suiveur d'une position courte.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // pointeur sur un objet CPositionInfo
   double&         sl,            // Prix du Stop Loss
   double&         tp             // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Le prix maximum du Stop Loss est d'abord calculé, le plus près du prix courant en utilisant les valeurs de la moyenne mobile de la barre précédente (barre complète).

Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est utilisé.

Si le prix calculé du Stop Loss est supérieur au prix de base et inférieur au prix minimum du Stop Loss autorisé, cela suggère de définir un nouveau prix de Stop Loss.

© 2000-2024, MetaQuotes Ltd.

CTrailingNone

La classe CTrailingNone est un stub. Cette classe devrait être utilisée au moment de l'initialisation de l'objet de Suivi si votre stratégie n'utilise pas de Stop Suiveur.

Description

La classe CTrailingNone n'implémente aucun algorithme de Stop Suiveur. Les méthodes de vérification des conditions des Stops Suiveurs retournent toujours faux.

Déclaration

```
class CTrailingNone: public CExpertTrailing
```

Titre

```
#include <Expert\Trailing\TrailingNone.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertTrailing
      CTrailingNone

Méthodes de Classe

Méthodes de Vérifications des Suivis	
virtual CheckTrailingStopLong	Une méthode stub pour vérifier les conditions du Stop Suiveur d'une position longue
virtual CheckTrailingStopShort	Une méthode stub pour vérifier les conditions du Stop Suiveur d'une position courte

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

Méthodes héritées de la classe CExpertTrailing

CheckTrailingStopLong, CheckTrailingStopShort

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Vérifie les conditions du Stop Suiveur d'une position longue.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // pointeur sur un objet CPositionInfo
   double&         sl,            // Prix du Stop Loss
   double&         tp             // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

La fonction retourne toujours faux.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Vérifie les conditions du Stop Suiveur d'une position courte.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // pointeur sur un objet CPositionInfo
   double&         sl,            // Prix du Stop Loss
   double&         tp             // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

La fonction retourne toujours faux.

© 2000-2024, MetaQuotes Ltd.

CTrailingPSAR

CTrailingPSAR est une classe d'implémentation de l'algorithme de Stop Suiveur basé sur les valeurs de l'indicateur Parabolic SAR.

Description

La classe CTrailingPSAR implémente l'algorithme de Stop Suiveur basé sur les valeurs de l'indicateur Parabolic SAR sur la barre précédente (complétée).

Déclaration

```
class CTrailingPSAR: public CExpertTrailing
```

Titre

```
#include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertTrailing
      CTrailingPSAR

Méthodes de Classe

Initialisation	
Step	Définit la valeur de pas de l'indicateur Parabolic SAR
Maximum	Définit la valeur maximum de l'indicateur Parabolic SAR
virtual InitIndicators	Initialise les indicateurs et les séries de données.
Méthodes de Vérifications des Suivis	
virtual CheckTrailingStopLong	Vérifie les conditions du Stop Suiveur d'une position longue
virtual CheckTrailingStopShort	Vérifie les conditions du Stop Suiveur d'une position courte

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Step

Définit la valeur de pas de l'indicateur Parabolic SAR.

```
void  Step(
   double  step      // Pas
   )
```

Paramètres

step

[in] La valeur du pas de l'indicateur Parabolic SAR.

© 2000-2024, MetaQuotes Ltd.

Maximum

Définit la valeur maximum de l'indicateur Parabolic SAR.

```
void  Maximum(
   double  maximum      // Maximum
   )
```

Paramètres

maximum

[in] La valeur du maximum de l'indicateur Parabolic SAR.

© 2000-2024, MetaQuotes Ltd.

InitIndicators

Initialise les indicateurs et les séries de données.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // Pointeur vers une collection de CIndicators
   )
```

Paramètres

indicators

[in] Pointeur vers une collection d'indicateurs et de séries de données (membre de la classe CExpert).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Vérifie les conditions du Stop Suiveur d'une position longue.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // pointeur
   double&         sl,            // Lien
   double&         tp             // Lien
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Le prix maximum du Stop Loss est d'abord calculé, le plus près du prix courant en utilisant les valeurs de l'indicateur Parabolic SAR de la barre précédente (barre complète).

Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est considérée comme le prix de base.

Si le prix calculé du Stop Loss est supérieur au prix de base et inférieur au prix maximum du Stop Loss autorisé, cela suggère de définir un nouveau prix de Stop Loss.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Vérifie les conditions du Stop Suiveur d'une position courte.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // pointeur
   double&         sl,            // Lien
   double&         tp             // Lien
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Le prix maximum du Stop Loss est d'abord calculé, le plus près du prix courant en utilisant les valeurs de l'indicateur Parabolic SAR de la barre précédente (barre complète).

Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est considérée comme le prix de base.

Si le prix calculé du Stop Loss est supérieur au prix de base et inférieur au prix minimum du Stop Loss autorisé, cela suggère de définir un nouveau prix de Stop Loss.

© 2000-2024, MetaQuotes Ltd.

Classes de Money Management

Cette section contient les détails techniques d'utilisation des classes de gestion du capital et du risque (money et risk management) et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création (et du test) de vos stratégies de trading.

La Bibliothèque Standard MQL5 (en terme de classes de money management et de risk management) est située dans le répertoire de travail du terminal dans le répertoire Include\Expert\Money\.

Classe	Description
CMoneyFixedLot	Cette classe implémente un algorithme de money management basé sur le trading avec une taille de lot fixe et prédéfinie.
CMoneyFixedMargin	Cette classe implémente un algorithme de money management basé sur le trading avec une marge fixe et prédéfinie.
CMoneyFixedRisk	Cette classe implémente un algorithme de money management basé sur le trading avec un risque prédéfini.
CMoneyNone	Cette classe implémente un algorithme de money management basé sur le trading une taille de lot minimum autorisée.
CMoneySizeOptimized	Cette classe implémente un algorithme de money management basé sur le trading avec une taille de lot variable, en fonction des résultats des deals précédents.

© 2000-2024, MetaQuotes Ltd.

CMoneyFixedLot

CMoneyFixedLot est la classe de l'algorithme de money management, basé sur le trading avec une taille de lot fixe.

Description

CMoneyFixedLot implémente l'algorithme de money management, basé sur le trading avec une taille de lot fixe.

Déclaration

```
class CMoneyFixedLot: public CExpertMoney
```

Titre

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedLot

Méthodes de Classe

Initialisation	
Lots	Définit le volume de trading
virtual ValidationSettings	Vérifie les paramètres
Méthodes de Money et de Risk Management (Gestion du Capital et du Risque)	
virtual CheckOpenLong	Retourne le volume de trading pour une position longue
virtual CheckOpenShort	Retourne le volume de trading pour une position courte

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Méthodes héritées de la classe CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Lots

Définit le volume de trading (en lots).

```
void  Lots(
   double  lots      // Lots
   )
```

Paramètres

lots

[in] Volume de trading (en lots).

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Vérifie le volume de trading spécifié.

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenLong(
   double  price,      // prix
   double  sl          // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne toujours le volume de trading fixé, défini par la méthode Lots.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Retourne le volume de trading pour une position courte.

```
virtual double  CheckOpenShort(
   double  price,      // prix
   double  sl          // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position courte.

Note

La fonction retourne toujours le volume de trading fixé, défini par la méthode Lots.

© 2000-2024, MetaQuotes Ltd.

CMoneyFixedMargin

CMoneyFixedMargin est la classe de l'algorithme de money management, basé sur le trading avec une marge fixe prédéfinie.

Description

CMoneyFixedMargin implément l'algorithme de money management, basé sur le trading avec une marge fixe prédéfinie.

Déclaration

```
class CMoneyFixedMargin: public CExpertMoney
```

Titre

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedMargin

Méthodes de Classe

Méthodes de Money et de Risk Management (Gestion du Capital et du Risque)	
virtual CheckOpenLong	Retourne le volume de trading pour une position longue
virtual CheckOpenShort	Retourne le volume de trading pour une position courte

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Méthodes héritées de la classe CExpertMoney

Percent, ValidationSettings, CheckReverse, CheckClose

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenLong(
   double  price,      // prix
   double  sl          // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne le volume de trading pour une position longue, elle utilise la marge fixe. La marge est défini par le paramètre "Percent" de la classe de base CExpertMoney.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Retourne le voume de trading pour une position courte.

```
virtual double  CheckOpenShort(
   double  price,      // prix
   double  sl          // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position courte.

Note

La fonction retourne le volume de trading pour une position courte, elle utilise la marge fixe. La marge est défini par le paramètre "Percent" de la classe de base CExpertMoney.

© 2000-2024, MetaQuotes Ltd.

CMoneyFixedRisk

CMoneyFixedRisk est la classe pour implémenter un algorithme de money management avec un risque fixe et prédéfini.

Description

La classe CMoneyFixedRisk implémente l'algorithme de money management avec un risque fixe et prédéfini.

Déclaration

```
class CMoneyFixedRisk: public CExpertMoney
```

Titre

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedRisk

Méthodes de Classe

Méthodes de Money et de Risk Management (Gestion du Capital et du Risque)	
virtual CheckOpenLong	Retourne le volume de trading pour une position longue
virtual CheckOpenShort	Retourne le volume de trading pour une position courte

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Méthodes héritées de la classe CExpertMoney

Percent, ValidationSettings, CheckReverse

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenLong(
   double  price,      // prix
   double  sl          // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne le volume de trading pour une position longue, elle utilise un risque fixe. Le risque est défini par le paramètre "Percent" de la classe de base CExpertMoney.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Retourne le volume de trading pour une position courte.

```
virtual double  CheckOpenShort(
   double  price,      // prix
   double  sl          // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position courte.

Note

La fonction retourne le volume de trading pour une position courte, elle utilise un risque fixe. Le risque est défini par le paramètre "Percent" de la classe de base CExpertMoney.

© 2000-2024, MetaQuotes Ltd.

CMoneyNone

CMoneyNone est une classe implémentant l'algorithme de trading avec un lot minimum autorisé.

Description

La classe CMoneyNone implémente le trading avec un lot minimum autorisé.

Déclaration

```
class CMoneyNone: public CExpertMoney
```

Titre

```
#include <Expert\Money\MoneyNone.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertMoney
      CMoneyNone

Méthodes de Classe

Initialisation	
virtual ValidationSettings	Vérifie les paramètres
Méthodes de Money et de Risk Management (Gestion du Capital et du Risque)	
virtual CheckOpenLong	Retourne le volume de trading pour une position longue
virtual CheckOpenShort	Retourne le volume de trading pour une position courte

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Méthodes héritées de la classe CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La fonction retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenLong(
   double  price,      // prix
   double  sl          // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne toujours la taille de lot minimale.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenShort(
   double  price,      // prix
   double  sl          // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position courte.

Note

La fonction retourne toujours la taille de lot minimale.

© 2000-2024, MetaQuotes Ltd.

CMoneySizeOptimized

CMoneySizeOptimized est une classe implémentant un algorithme de money management, basé sur le trading avec taille variable de lot, suivant les résultats des deals précédents.

Description

CMoneySizeOptimized implémenté un algorithme de money management, basé sur le trading avec une taille variable de lot, suivant les résultats des deals précédents.

Déclaration

```
class CMoneySizeOptimized: public CExpertMoney
```

Titre

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

Hiérarchie d'héritage

CObject
  CExpertBase
    CExpertMoney
      CMoneySizeOptimized

Méthodes de Classe

Initialisation	
DecreaseFactor	Définit la valeur du facteur de réduction
virtual ValidationSettings	Vérifie les paramètres
Méthodes de Money et de Risk Management (Gestion du Capital et du Risque)	
virtual CheckOpenLong	Retourne le volume de trading pour une position longue
virtual CheckOpenShort	Retourne le volume de trading pour une position courte

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Méthodes héritées de la classe CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

DecreaseFactor

Définit la valeur du facteur de réduction.

```
void  DecreaseFactor(
   double  decrease_factor      // Facteur de réduction
   )
```

Paramètres

decrease_factor

[in] Facteur de réduction.

Note

Le DecreaseFactor définit le coefficient de réduction du volume (comparé au volume de la position précédente) dans le cas de positions perdantes consécutives.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si la valeur du facteur de réduction est négative, retourne faux, sinon retourne vrai.

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenLong(
   double  price,      // prix
   double  sl          // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne le volume de trading pour une position longue, le volume dépendant des résultats des deals précédents.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Retourne le volume de trading pour une position courte.

```
virtual double  CheckOpenShort(
   double  price,      // prix
   double  sl          // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne le volume de trading pour une position courte, le volume dépendant des résultats des deals précédents.

© 2000-2024, MetaQuotes Ltd.

Classes de Création des Panneaux de Contrôle et des Dialogues

Cette section contient les détails techniques d'utilisation des classes de création des panneaux de contrôle et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création des panneaux de contrôle dans vos programmes MQL5 (Expert Advisors et indicateurs).

La Bibliothèque Standard MQL5 (en terme de contrôles) est située dans le répertoire de travail du terminal dans le répertoire Include\Controls.

Trouvez les exemples d'utilisation des classes dans les articles suivants :

- How to create a graphical panel of any complexity level
- Improving Panels: Adding transparency, changing background color and inheriting from CAppDialog/CWndClient
- Adding a control panel to an indicator or an Expert Advisor in no time
- Create your own graphical panels in MQL5
- Creating active control panels in MQL5 for trading

Des exemples d'Expert Advisor illustrant l'utilisation des ces classes peuvent être trouvés dans le répertoire MQL5\Expert\Examples\Controls.

Structure auxiliaires	Description
CRect	Structure représentant une zone rectangulaire
CDateTime	Structure permettant de travailler avec la date et l'heure

Classes de base	Description
CWnd	Classe de base pour tous les contrôles
CWndObj	Classe de base pour les contrôles et les dialogues
CWndContainer	Classe de base pour les contrôles complexes (contenant des contrôles inter-dépendants)

Contrôles Simples	Description
CLabel	Contrôle, basé sur l'objet graphique "Etiquette de Texte"
CBmpButton	Contrôle, basé sur l'objet graphique "Etiquette Bitmap"
CButton	Contrôle, basé sur l'objet graphique "Bouton"

© 2000-2024, MetaQuotes Ltd.

Contrôles Simples	Description
CEdit	Contrôle, basé sur l'objet graphique "Champ d'édition"
CPanel	Contrôle, basé sur l'objet graphique "Etiquette rectangulaire"
CPicture	Contrôle, basé sur l'objet graphique "Etiquette bitmap"

Contrôles complexes	Description
CScroll	Classe de base des barres de défilement
CScrollV	Barre de défilement verticale
CScrollH	Barre de défilement horizontale
CWndClient	Classe de base d'une zone client avec barres de défilement
CListView	Liste
CComboBox	Liste déroulante
CCheckBox	Case à cocher
CCheckGroup	Groupe de cases à cocher
CRadioButton	Bouton radio
CRadioGroup	Groupe de boutons radio
CSpinEdit	Compteur
CDialog	Dialogue
CAppDialog	Dialogue d'application

© 2000-2024, MetaQuotes Ltd.

CRect

La classe CRect représente la zone rectangulaire du graphique.

Description

La classe CRect est la classe de la zone graphique, elle est définie par les coordonnées des coins supérieur gauche et inférieur droit d'un rectangle en coordonnées Cartésiennes.

Déclaration

```
class CRect
```

Titre

```
#include <Controls\Rect.mqh>
```

Méthodes de Classe

Propriétés	
Left	Retourne/Définit la coordonnée X du coin supérieur gauche
Top	Retourne/Définit la coordonnée Y du coin supérieur gauche
Right	Retourne/Définit la coordonnée X du coin inférieur droit
Bottom	Retourne/Définit la coordonnée Y du coin inférieur droit
Width	Retourne/Définit la largeur
Height	Retourne/Définit la hauteur
SetBound	Définit les nouvelles coordonnées en utilisant la classe CRect
Move	Définit les nouvelles coordonnées de la classe CRect
Shift	Effectue le décalage relatif des coordonnées de la classe CRect
Contains	Vérifie si le point est situé à l'intérieur de la zone de la classe CRect
Méthodes supplémentaires	
Format	Retourne les coordonnées de la zone sous forme d'une chaîne de caractères

© 2000-2024, MetaQuotes Ltd.

Left (Méthode "Get")

Retourne la coordonnée X du coin supérieur gauche.

```
int  Left()
```

Valeur de retour

Coordonnée X du coin supérieur gauche.

Left (Méthode "Set")

Définit la coordonnée X du coin supérieur gauche.

```
void  Left(
   const int  x      // nouvelle coordonnée X
   )
```

Paramètres

x

[in] Nouvelle coordonnée X du coin supérieur gauche.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Top (Méthode "Get")

Retourne la coordonnée Y du coin supérieur gauche.

```
int  Top()
```

Valeur de retour

Coordonnée Y du coin supérieur gauche.

Top (Méthode "Set")

Définit la nouvelle coordonnée Y du coin supérieur gauche.

```
void  Top(
   const int  y      // Coordonnée Y
   )
```

Paramètres

y

[in] Nouvelle coordonnée Y du coin supérieur gauche.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Right (Méthode "Get")

Retourne la coordonnée X du coin inférieur droit.

```
int  Right()
```

Valeur de retour

Coordonnée X du coin inférieur droit.

Right (Méthode "Set")

Définit la coordonnée Y du coin inférieur droit.

```
void  Right(
   const int  x      // coordonnée x
   )
```

Paramètres

x

[in] Nouvelle coordonnée X du coin inférieur droit.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Bottom (Méthode "Get")

Retourne la coordonnée Y du coin inférieur droit.

```
int  Bottom()
```

Valeur de retour

Coordonnée Y du coin inférieur droit.

Bottom (Méthode "Set")

Définit la coordonnée Y du coin inférieur droit.

```
void  Bottom(
   const int  y      // Coordonnée Y
   )
```

Paramètres

y

[in] Nouvelle coordonnée Y du coin inférieur droit.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Width (Méthode "Get")

Retourne la largeur de la zone.

```
int  Width()
```

Valeur de retour

Largeur de la zone.

Width (Méthode "Set")

Définit la nouvelle largeur de la zone.

```
virtual bool  Width(
   const int  w      // largeur
   )
```

Paramètres

w

[in] Nouvelle largeur.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Height (Méthode "Get")

Retourne la hauteur de la zone.

```
int  Height()
```

Valeur de retour

Hauteur de la zone.

Height (Méthode "Set")

Définit la nouvelle hauteur de la zone.

```
virtual bool  Height(
   const int  h      // hauteur
   )
```

Paramètres

h

[in] Nouvelle hauteur.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

SetBound

Définit les nouvelles coordonnées de la zone utilisant les coordonnées de la classe CRect.

```
void  SetBound(
   const & CRect  rect      // Classe CRect
   )
```

Valeur de retour

Aucune.

SetBound

Définit les nouvelles coordonnées de la zone.

```
void  SetBound(
   const int  l      // ghauche
   const int  t      // dessus
   const int  r      // droit
   const int  b      // dessous
   )
```

Paramètres

l

[in] Coordonnée X du coin supérieur gauche.

t

[in] Coordonnée Y du coin supérieur gauche.

r

[in] Coordonnée X du coin inférieur droit.

b

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Move

Définit les nouvelles coordonnées de la classe CRect.

```
void  Move(
   const int  x,      // coordonnée X
   const int  y       // coordonnée Y
   )
```

Paramètres

x

[in] Nouvelle coordonnée X.

y

[in] Nouvelle coordonnée Y.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Shift

Effectue le décalage relatif des coordonnées de la classe CRect.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Paramètres

dx

[in] Delta X.

dy

[in] Delta Y.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Contains

Vérifie si le point est situé à l'intérieur de la zone de la classe CRect.

```
bool  Contains(
   const int  x,      // coordonnée X
   const int  y       // coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X.

y

[in] Coordonnée Y.

Valeur de retour

vrai si le point est situé à l'intérieur de la zone (incluant les bordures), sinon faux.

© 2000-2024, MetaQuotes Ltd.

Format

Retourne les coordonnées de la zone sous forme d'une chaîne de caractères.

```
string  Format(
   string &  fmt,     // format
   ) const
```

Paramètres

fmt

[in] Chaîne de caractères avec le format.

Valeur de retour

Chaîne de caractères avec les coordonnées de la zone.

© 2000-2024, MetaQuotes Ltd.

CDateTime

La classe CDateTime permet de travailler avec les dates et heures.

Description

CDateTime est une structure dérivant de MqlDateTime, elle est utilisée pour toutes les opérations utilisant des dates et des heures dans les contrôles.

Déclaration

```
struct CDateTime
```

Titre

```
#include <Tools\DateTime.mqh>
```

Méthodes de Classe

Propriétés	
MonthName	Retourne le nom du mois
ShortMonthName	Retourne le nom court du mois
DayName	Retourne le nom du jour de la semaine
ShortDayName	Retourne le nom court du jour de la semaine
DaysInMonth	Retourne le nombre de jours dans le mois
Méthodes Get/Set	
DateTime	Retourne/Définit la date/heure
Date	Définit la date
Time	Définit l'heure
Sec	Définit les secondes
Min	Définit les minutes
Hour	Définit les heures
Day	Définit le jour du mois
Mon	Définit le mois
Year	Définit l'année
Méthodes supplémentaires	
SecDec	Soustrait le nombre de secondes
SecInc	Ajoute le nombre de secondes
MinDec	Soustrait le nombre spécifié de minutes

© 2000-2024, MetaQuotes Ltd.

Propriétés	
MinInc	Ajoute le nombre spécifié de minutes
HourDec	Soustrait le nombre spécifié d'heures
HourInc	Ajoute le nombre spécifié d'heures
DayDec	Soustrait le nombre spécifié de jours
DayInc	Ajoute le nombre spécifié de jours
MonDec	Soustrait le nombre spécifié de mois
MonInc	Ajoute le nombre spécifié de mois
YearDec	Soustrait le nombre spécifié d'années
YearInc	Ajoute le nombre spécifié d'années

© 2000-2024, MetaQuotes Ltd.

MonthName

Retourne le nom du mois.

```
string  MonthName()  const
```

Retourne le nom du mois par son index.

```
string  MonthName(
   const int      num         // index du mois
   )  const
```

Paramètres

num

[in] Index du mois (1-12).

Valeur de retour

Nom du mois.

© 2000-2024, MetaQuotes Ltd.

ShortMonthName

Retourne le nom court du mois.

```
string  ShortMonthName()  const
```

Retourne le nom court du mois par son index.

```
string  ShortMonthName(
   const int      num         // index du mois
   )  const
```

Paramètres

num

[in] Index du mois (1-12).

Valeur de retour

Nom court du mois.

© 2000-2024, MetaQuotes Ltd.

DayName

Retourne le nom du jour de la semaine.

```
string  DayName()  const
```

Retourne le nom du jour de la semaine par son index.

```
string  DayName(
   const int      num         // index du jour
   )  const
```

Paramètres

num

[in] Index du jour (de 0 à 6).

Valeur de retour

Nom du jour.

© 2000-2024, MetaQuotes Ltd.

ShortDayName

Retourne le nom court du jour de la semaine.

```
string  ShortDayName()  const
```

Retourne le nom du jour de la semaine par son index..

```
string  ShortDayName(
   const int      num         // index du jour
   )  const
```

Paramètres

num

[in] Index du jour (de 0 à 6).

Valeur de retour

Nom court du jour.

© 2000-2024, MetaQuotes Ltd.

DaysInMonth

Retourne le nombre de jours dans le mois

```
int DaysInMonth() const
```

Valeur de retour

Nombre de jours dans le mois

© 2000-2024, MetaQuotes Ltd.

DateTime (Méthode Get)

Retourne la date et l'heure.

```
datetime  DateTime()
```

Valeur de retour

Valeur du type datetime.

DateTime (Méthode Set avec datetime)

Définit la date et l'heure à partir du type datetime.

```
void  DateTime(
   const datetime      value         // date et heure
   )
```

Paramètres

value

[in] Valeur du type datetime.

Valeur de retour

Aucune.

DateTime (Méthode Set avec MqlDateTime)

Définit la date et l'heure à partir du type MqlDateTime.

```
void  DateTime(
   const MqlDateTime   &value        // date et heure
   )
```

Paramètres

value

[in] Valeur du type MqlDateTime.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Date (Méthode Set avec datetime)

Définit la date à partir du type datetime.

```
void  Date(
   const datetime      value         // date
   )
```

Paramètres

value

[in] Valeur du type datetime.

Valeur de retour

Aucune.

Date (Méthode Set avec MqlDateTime)

Définit la date à partir du type MqlDateTime.

```
void  Date(
   const MqlDateTime   &value        // date
   )
```

Paramètres

value

[in] Valeur du type MqlDateTime.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Time (Méthode Set avec datetime)

Définit l'heure à partir du type datetime.

```
void  Time(
   const datetime      value         // heure
   )
```

Paramètres

value

[in] Valeur du type datetime.

Valeur de retour

Aucune.

Time (Méthode Set avec MqlDateTime)

Définit l'heure à partir du type MqlDateTime.

```
void  Time(
   const MqlDateTime   &value        // heure
   )
```

Paramètres

value

[in] Valeur du type MqlDateTime.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Sec

Définit les secondes.

```
void  Sec(
   const int   value        // secondes
   )
```

Paramètres

value

[in] Nombre de secondes.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Min

Définit le nombre de minutes.

```
void  Min(
   const int   value        // minutes
   )
```

Paramètres

value

[in] Nombre de minutes.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Hour

Définit l'heure.

```
void  Hour(
   const int   value        // heure
   )
```

Paramètres

value

[in] Heure.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Day

Définit le jour du mois

```
void  Day(
   const int   value        // jour
   )
```

Paramètres

value

[in] Jour du mois.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Mon

Définit le mois.

```
void  Mon(
   const int   value        // mois
   )
```

Paramètres

value

[in] Numéro du mois.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Year

Définit l'année.

```
void  Year(
   const int   value        // année
   )
```

Paramètres

value

[in] Année.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

SecDec

Soustrait le nombre de secondes spécifiées.

```
void  SecDec(
   int   delta=1        // secondes
   )
```

Paramètres

delta

[in] Nombre de secondes à soustraire.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

SecInc

Ajoute le nombre spécifié de secondes

```
void  SecInc(
   int   delta=1        // secondes
   )
```

Paramètres

delta

[in] Nombre de secondes à ajouter.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

MinDec

Soustrait le nombre spécifié de minutes.

```
void  MinDec(
   int   delta=1        // minutes
   )
```

Paramètres

delta

[in] Nombre de minutes à soustraire.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

MinInc

Ajoute le nombre spécifié de minutes.

```
void  MinInc(
   int   delta=1        // minutes
   )
```

Paramètres

delta

[in] Nombre de minutes à ajouter.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

HourDec

Soustrait le nombre spécifié d'heures.

```
void  HourDec(
   int   delta=1        // heures
   )
```

Paramètres

delta

[in] Nombre d'heures à soustraire.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

HourInc

Ajoute le nombre spécifié d'heures.

```
void  HourInc(
   int   delta=1        // heures
   )
```

Paramètres

delta

[in] Nombre d'heures à ajouter.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

DayDec

Soustrait le nombre spécifié de jours.

```
void  DayDec(
   int   delta=1        // jours
   )
```

Paramètres

delta

[in] Nombre de jours à soustraire.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

DayInc

Ajoute le nombre spécifié de jours.

```
void  DayInc(
   int   delta=1        // jours
   )
```

Paramètres

delta

[in] nombre de jours à ajouter.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

MonDec

Soustrait le nombre spécifié de mois.

```
void  MonDec(
   int   delta=1        // mois
   )
```

Paramètres

delta

[in] Nombre de mois à soustraire.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

MonInc

Ajoute le nombre spécifié de mois.

```
void  MonInc(
   int   delta=1        // mois
   )
```

Paramètres

delta

[in] Nombre de mois à ajouter.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

YearDec

Soustrait le nombre spécifié d'années.

```
void  YearDec(
   int   delta=1        // années
   )
```

Paramètres

delta

[in] Nombre d"années à soustraire.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

YearInc

Ajoute le nombre spécifié d'années.

```
void  YearInc(
   int   delta=1        // années
   )
```

Paramètres

delta

[in] Nombre d'années à ajouter.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CWnd

La classe CWnd est la classe de base de tous les contrôles inclus dans la Bilbiothèque Standard MQL5 (MQL5 Standard Library).

Description

La classe CWnd est l'implémentation de la classe du contrôle de base.

Déclaration

```
class CWnd : public CObject
```

Titre

```
#include <Controls\Wnd.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd

Descendants directs

CDragWnd, CWndContainer, CWndObj

Méthodes de Classe

Création et destruction	
Create	Crée le contrôle
Destroy	Détruit le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
OnMouseEvent	Gestionnaire d'évènement de l'évènement CHARTEVENT_MOUSE_MOVE
Nom	
Name	Retourne le nom du contrôle
Accès au conteneur	
ControlsTotal	Retourne le nombre total de contrôles du conteneur
Control	Retourne le contrôle par son index
ControlFind	Retourne le contrôle par son identifiant
Géometrie	

© 2000-2024, MetaQuotes Ltd.

Création et destruction	
Rect	Retourne le pointeur sur l'objet CRect
Left	Retourne/Définit la coordonnée X du coin supérieur gauche
Top	Retourne/Définit la coordonnée Y du coin supérieur gauche
Right	Retourne/Définit la coordonnée X du coin inférieur droit
Bottom	Retourne/Définit la coordonnée Y du coin inférieur droit
Width	Retourne/Définit la largeur
Height	Retourne/Définit la hauteur
Move	Définit les nouvelles coordonnées du point.
Shift	Effectue le décalage relatif des coordonnées du contrôle
Resize	Définit les nouvelles largeur/hauteur du contrôle
Contains	Vérifie si le point est situé à l'intérieur de la zone du contrôle
Alignement	
Alignment	Définit les propriétés d'alignement du contrôle
Align	Effectue l'alignement du contrôle
Identification	
Id	Retourne/Définit l'identifiant du contrôle
Etat	
IsEnabled	Retourne/Définit une valeur indiquant si le contrôle est activé ou pas
Enable	Définit une valeur indiquant si le contrôle est activé ou pas
Disable	Désactive le contrôle
IsVisible	Vérifie la visibilité du contrôle
Visible	Définit la visibilité du contrôle
Show	Affiche le contrôle
Hide	Cache le contrôle
IsActive	Vérifie si le contrôl est actif ou pas
Activate	Active le contrôle
Deactivate	Désactive le contrôle
Flags d'états	
StateFlags	Retourne/Définit les flags d'état du contrôle
StateFlagsSet	Définit les flags d'état du contrôle

© 2000-2024, MetaQuotes Ltd.

Création et destruction	
StateFlagsReset	Réinitialise les flags d'état du contrôle
Flags des propriétés	
PropFlags	Retourne/Définit les flags des propriétés du contrôle
PropFlagsSet	Définit les flags des propriétés du contrôle
PropFlagsReset	Réinitialise les flags des propriétés du contrôle
HandlerOpérations de la souris	
MouseX	Retourne/Définit la coordonnée X de la souris
MouseY	Retourne/Définit la coordonnée Y de la souris
MouseFlags	Retourne/Définit l'état des boutons de la souris
MouseFocusKill	Supprime le focus de la souris
Gestionnaires d'évènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnDestroy	Gestionnaire d'évènement "Destroy"
OnMove	Gestionnaire d'évènement "Move"
OnResize	Gestionnaire d'évènement "Resize"
OnEnable	Gestionnaire d'évènement "Enable"
OnDisable	Gestionnaire d'évènement "Disable"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnActivate	Gestionnaire d'évènement "Activate"
OnDeactivate	Gestionnaire d'évènement "Deactivate"
OnClick	Gestionnaire d'évènement "Click"
OnChange	Gestionnaire d'évènement "Change"
Gestionnaires d'évènement de la souris	
OnMouseDown	Gestionnaire d'évènement "MouseDown"
OnMouseUp	Gestionnaire d'évènement "MouseUp"
Gestionnaires d'évènement de Drag n' Drop	
OnDragStart	Gestionnaire d'évènement "DragStart"

© 2000-2024, MetaQuotes Ltd.

Création et destruction	
OnDragProcess	Gestionnaire d'évènement "DragProcess"
OnDragEnd	Gestionnaire d'évènement "DragEnd"
Objet de déplacement	
DragObjectCreate	Crée un objet de déplacement
DragObjectDestroy	Détruit un objet de déplacement

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Create

Crée le contrôle.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La méthode de la classe de base ne fait que sauvegarder les paramètres et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

Destroy

Détruit un contrôle.

```
virtual bool  Destroy()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre de l'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre de l'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnMouseEvent

Gestionnaire d'évènement de la souris (l'évènement CHARTEVENT_MOUSE_MOVE).

```
virtual bool  OnMouseEvent(
   const int  x,          // coordonnée X
   const int  y,          // coordonnée Y
   const int  flags       // flags
   )
```

Paramètres

x

[in] Coordonnée X du curseur de la souris relativement au coin supérieur gauche du graphique.

y

[in] Coordonnée Y du curseur de la souris relativement au coin supérieur gauche du graphique.

flags

[in] Flag des états des boutons de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

Name

Retourne le nom du contrôle

```
string  Name()  const
```

Valeur de retour

Nom du contrôle.

© 2000-2024, MetaQuotes Ltd.

ControlsTotal

Retourne le nombre total de contrôles du conteneur.

```
int ControlsTotal() const
```

Valeur de retour

Nombre de contrôles du conteneur.

Note

La méthode de la classe de base ne possède pas le conteneur, elle fournit un accès au conteneur à ses descendants et retourne toujours 0.

© 2000-2024, MetaQuotes Ltd.

Control

Retourne le contrôle par son index

```
CWnd*  Control(
   const int  ind      // index
   )  const
```

Paramètres

ind

[in] Index du contrôle.

Valeur de retour

Un pointeur sur le contrôle.

Note

La méthode de la classe de base ne possède pas le conteneur, elle fournit un accès au conteneur à ses descendants et retourne toujours NULL.

© 2000-2024, MetaQuotes Ltd.

ControlFind

Retourne le contrôle par son identifiant spécifié

```
virtual CWnd*  ControlFind(
   const long  id      // Identifiant
   )
```

Paramètres

id

[in] Identifiant du contrôle à trouver.

Valeur de retour

Pointeur sur le contrôle.

Note

La méthode de la classe de base ne possède pas le conteneur, elle fournit un accès au conteneur à ses descendants. Si l'identifiant spécifié correspond à l'identifiant du conteneur, elle retourne un pointeur sur elle-même (this).

© 2000-2024, MetaQuotes Ltd.

Rect

Retourne le pointeur sur l'objet CRect.

```
const CRect*  Rect()  const
```

Valeur de retour

Retourne le pointeur sur l'objet CRect.

© 2000-2024, MetaQuotes Ltd.

Left (Méthode "Get")

Retourne la coordonnée X du coin supérieur gauche du contrôle.

```
int  Left()
```

Valeur de retour

Coordonnée X du coin supérieur gauche du contrôle.

Left (Méthode "Set")

Définit la coordonnée X du coin supérieur gauche du contrôle.

```
void  Left(
   const int  x      // nouvelle coordonnée X
   )
```

Paramètres

x

[in] Nouvelle coordonnée X du coin supérieur gauche.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Top (Méthode "Get")

Retourne la coordonnée Y du coin supérieur gauche du contrôle.

```
int  Top()
```

Valeur de retour

Coordonnée Y du coin supérieur gauche du contrôle.

Top (Méthode "Set")

Définit la coordonnée Y du coin supérieur gauche du contrôle.

```
void  Top(
   const int  y      // Coordonnée Y
   )
```

Paramètres

y

[in] Nouvelle coordonnée Y du coin supérieur gauche.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Right (Méthode "Get")

Retourne la coordonnée Y du coin inférieur droit du contrôle.

```
int  Right()
```

Valeur de retour

Coordonnée X du coin inférieur droit.

Right (Méthode "Set")

Définit la coordonnée Y du coin inférieur droit du contrôle.

```
void  Right(
   const int  x      // coordonnée x
   )
```

Paramètres

x

[in] Nouvelle coordonnée X du coin inférieur droit.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Bottom (Méthode "Get")

Retourne la coordonnée Y du coin inférieur droit du contrôle.

```
int  Bottom()
```

Valeur de retour

La coordonnée Y du coin inférieur droit du contrôle.

Bottom (Méthode "Set")

Définit la coordonnée Y du coin inférieur droit du contrôle.

```
void  Bottom(
   const int  y      // Coordonnée Y
   )
```

Paramètres

y

[in] Nouvelle coordonnée Y du coin inférieur droit.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Width (Méthode "Get")

Retourne la largeur du contrôle.

```
int  Width()
```

Valeur de retour

Largeur du contrôle.

Width (Méthode "Set")

Définit la nouvelle largeur du contrôle.

```
virtual bool  Width(
   const int  w      // largeur
   )
```

Paramètres

w

[in] Nouvelle largeur.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Height (Méthode "Get")

Retourne la hauteur du contrôle.

```
int  Height()
```

Valeur de retour

Hauteur du contrôle.

Height (Méthode "Set")

Définit la nouvelle hauteur du contrôle.

```
virtual bool  Height(
   const int  h      // hauteur
   )
```

Paramètres

h

[in] Nouvelle hauteur.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Move

Définit les nouvelles coordonnées du contrôle.

```
void  Move(
   const int  x,      // coordonnée X
   const int  y       // coordonnée Y
   )
```

Paramètres

x

[in] Nouvelle coordonnée X.

y

[in] Nouvelle coordonnée Y.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Shift

Effectue le décalage relatif des coordonnées du contrôle.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Paramètres

dx

[in] Delta X.

dy

[in] Delta Y.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Resize

Définit les nouvelles largeur/hauteur du contrôle.

```
virtual bool  Resize(
   const int  w,      // largeur
   const int  h       // hauteur
   )
```

Paramètres

w

[in] Nouvelle largeur.

h

[in] Nouvelle hauteur.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Contains

Vérifie si le point est situé à l'intérieur de la zone du contrôle.

```
bool  Contains(
   const int  x,      // coordonnée X
   const int  y       // coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X.

y

[in] Coordonnée Y.

Valeur de retour

vrai si le point est situé à l'intérieur de la zone (incluant les bordures), sinon faux.

Contains

Vérifie si le point est situé à l'intérieur de la zone du contrôle.

```
bool  Contains(
   const CWnd*  control      // pointeur
   )  const
```

Paramètres

control

[in] Pointeur sur l'objet.

Valeur de retour

vrai si le point est situé à l'intérieur de la zone (incluant les bordures), sinon faux.

© 2000-2024, MetaQuotes Ltd.

Alignment

Définit les paramètres d'alignement du contrôle.

```
void  Alignment(
   const int  flags,       // flags d'alignement
   const int  left,        // décalage depuis la bordure gauche
   const int  top,         // décalage depuis la bordure supérieure
   const int  right,       // décalage depuis la bordure droite
   const int  bottom       // décalage depuis la bordure inférieure
   )
```

Paramètres

flags

[in] Flags d'alignement.

left

[in] Décalage fixé depuis la bordure gauche.

top

[in] Décalage fixé depuis la bordure supérieure.

right

[in] Décalage fixé depuis la bordure droite.

bottom

[in] Décalage fixé depuis la bordure inférieure.

Valeur de retour

Aucune.

Note

Flags d'alignement :

```
enum WND_ALIGN_FLAGS
{
   WND_ALIGN_NONE=0,                                  // aucun alignement
   WND_ALIGN_LEFT=1,                                  // alignement à gauche
   WND_ALIGN_TOP=2,                                   // alignement en haut
   WND_ALIGN_RIGHT=4,                                 // alignement à droite
   WND_ALIGN_BOTTOM=8,                                // alignement en bas
   WND_ALIGN_WIDTH = WND_ALIGN_LEFT|WND_ALIGN_RIGHT,  // aligner la largeur
   WND_ALIGN_HEIGHT=WND_ALIGN_TOP|WND_ALIGN_BOTTOM,   // aligner la hauteur
   WND_ALIGN_CLIENT=WND_ALIGN_WIDTH|WND_ALIGN_HEIGHT, // aligner la hauteur et la larg
   }
```

© 2000-2024, MetaQuotes Ltd.

Align

Effectue l'alignement du contrôle dans la zone du graphique spécifiée.

```
virtual bool  Align(
   const CRect*  rect      // pointeur
   )
```

Paramètres

rect

[in] Pointeur vers l'objet avec les coordonnées de la zone du graphique.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Les paramètres d'alignement doivent être spécifiés (aucun alignement par défaut).

© 2000-2024, MetaQuotes Ltd.

Id (Méthode "Get")

Retourne l'identifiant du contrôle.

```
long Id() const
```

Valeur de retour

L'identifiant du contrôle.

Id (Méthode "Set")

Définit la nouvelle valeur de l'identifant du contrôle.

```
virtual long Id(
   const long id      // identifiant
   )
```

Paramètres

id

[in] Nouvelle valeur de l'identifiant du contrôle.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

IsEnabled

Retourne une valeur indiquant si le contrôle est activé ou pas

```
bool IsEnabled() const
```

Valeur de retour

vrai si le contrôle est actif, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Enable

Active le contrôle.

```
virtual bool  Enable()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si le contrôle est activé, il est capable de gérer les évènements externes.

© 2000-2024, MetaQuotes Ltd.

Disable

Désactive le contrôle.

```
virtual bool  Disable()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Un contrôle désactivé ne permet pas de traiter les évènements externes.

© 2000-2024, MetaQuotes Ltd.

IsVisible

Retourne une valeur indiquant si le contrôle est visible ou pas

```
bool IsVisible() const
```

Valeur de retour

vrai si le contrôle est affiché sur le graphique, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Visible

Définit la visibilité du contrôle.

```
virtual bool  Visible(
   const bool  flag      // flag
   )
```

Paramètres

flag

[in] Nouveaux flag de visibilité.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Show

Affiche le contrôle.

```
virtual bool  Show()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Hide

Cache le contrôle.

```
virtual bool  Hide()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

IsActive

Retourne une valeur indiquant si le contrôle est activé ou pas.

```
bool IsActive() const
```

Valeur de retour

vrai si le contrôle est actif, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Activate

Active le contrôle.

```
virtual bool  Activate()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Le contrôle devient actif lorsque le curseur de la souris passe au-dessus.

© 2000-2024, MetaQuotes Ltd.

Deactivate

Désactive le contrôle

```
virtual bool  Deactivate()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Le contrôle devient inactif lorsque le curseur de la souris sort de la zone du contrôle.

© 2000-2024, MetaQuotes Ltd.

StateFlags (Méthode "Get")

Retourne les flags d'état du contrôle

```
int  StateFlags()
```

Valeur de retour

Les flags d'état du contrôle.

StateFlags (Méthode "Set")

Définit les flags d'état du contrôle.

```
virtual void  StateFlags(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Nouveaux flags d'état du contrôle.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

StateFlagsSet

Définit les flags d'état du contrôle.

```
virtual void  StateFlagsSet(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Flags à définir (masque de bits).

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

StateFlagsReset

Réinitialise les flags d'état du contrôle.

```
virtual void  StateFlagsReset(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Flags à réinitialiser (masque de bits).

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

PropFlags (Méthode "Get")

Retourne les flags des propriétés du contrôle

```
void  PropFlags(
   const int  flags      // flags
   )
```

Valeur de retour

Les flags des propriétés du contrôle

PropFlags (Méthode "Set")

Définit les flags des propriétés du contrôle.

```
virtual void  PropFlags(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Nouveaux flags.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

PropFlagsSet

Définit les flags des propriétés du contrôle.

```
virtual void  PropFlagsSet(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Flags à définir (masque de bits).

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

PropFlagsReset

Réinitialise les flags des propriétés du contrôle.

```
virtual void  PropFlagsReset(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Flags à réinitialiser (masque de bits).

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

MouseX (Méthode "Set")

Sauvegarde la coordonnée X de la souris.

```
void  MouseX(
   const int  value      // coordonnée
   )
```

Paramètres

value

[in] La coordonnée X de la souris.

Valeur de retour

Aucune.

MouseX (Méthode "Get")

Retourne la coordonnée X sauvegardée de la souris.

```
int  MouseX()
```

Valeur de retour

La coordonnée X sauvegardée de la souris.

© 2000-2024, MetaQuotes Ltd.

MouseY (Méthode "Set")

Sauvegarde la coordonnée Y de la souris.

```
void  MouseY(
   const int  value      // coordonnée
   )
```

Paramètres

value

[in] La coordonnée Y de la souris.

Valeur de retour

Aucune.

MouseY (Méthode "Get")

Retourne la coordonnée Y sauvegardée de la souris.

```
int  MouseY()
```

Valeur de retour

La coordonnée Y sauvegardée de la souris.

© 2000-2024, MetaQuotes Ltd.

MouseFlags (Méthode "Set")

Sauvegarde l'état des boutons de la souris.

```
virtual void  MouseFlags(
   const int  value      // état
   )
```

Paramètres

value

[in] Etat des boutons de la souris.

Valeur de retour

Aucune.

MouseFlags (Méthode "Get")

Retourne l'état sauvegardé des boutons de la souris.

```
int  MouseFlags()
```

Valeur de retour

Etat des boutons de la souris.

© 2000-2024, MetaQuotes Ltd.

MouseFocusKill

Réinitialise l'état sauvegardé des boutons de la souris et désactive le contrôle.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // identifiant
   )
```

Paramètres

id=CONTROLS_INVALID_ID

[in] Identifiant du contrôle ayant reçu le focus de la souris.

Valeur de retour

Le résultat de la désactivation du contrôle.

© 2000-2024, MetaQuotes Ltd.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnDestroy

Le gestionnaire d'évènement "Destroy" du contrôle.

```
virtual bool  OnDestroy()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnEnable

Le gestionnaire d'évènement "Enable" (si activé, il peut répondre à l'interaction utilisateur) du contrôle.

```
virtual bool  OnEnable()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnDisable

Le gestionnaire d'évènement "Disable" (si désactivé, ne peut pas répondre à l'interaction utilisateur) du contrôle.

```
virtual bool  OnDisable()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnActivate

Le gestionnaire d'évènement "Activate" du contrôle.

```
virtual bool  OnActivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnDeactivate

Le gestionnaire d'évènement "Deactivate" du contrôle.

```
virtual bool  OnDeactivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnClick

Le gestionnaire d'évènement "Click" (clic avec le bouton gauche de la souris) du contrôle.

```
virtual bool  OnClick()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnMouseDown

Le gestionnaire d'évènement "MouseDown" du contrôle.

```
virtual bool  OnMouseDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseDown" est généré lorsque le bouton gauche de la souris est appuyé sur le contrôle.

© 2000-2024, MetaQuotes Ltd.

OnMouseUp

Le gestionnaire d'évènement "MouseUp" du contrôle (le bouton gauche de la souris est relâché).

```
virtual bool  OnMouseUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseUp" est généré lorsque le bouton gauche de la souris est relâché au-dessus du contrôle.

© 2000-2024, MetaQuotes Ltd.

OnDragStart

Le gestionnaire d'évènement "DragStart" du contrôle.

```
virtual bool  OnDragStart()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DragStart" est généré au démarrage du déplacement.

© 2000-2024, MetaQuotes Ltd.

OnDragProcess

Le gestionnaire d'évènement "DragProcess" du contrôle.

```
virtual bool  OnDragProcess(
   const int  x,      // Coordonnée X
   const int  y       // Coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X actuelle du curseur de la souris.

y

[in] Coordonnée Y actuelle du curseur de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DragProcess" est généré lorsque le contrôle est déplacé.

© 2000-2024, MetaQuotes Ltd.

OnDragEnd

Le gestionnaire d'évènement "DragEnd" du contrôle.

```
virtual bool  OnDragEnd()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DragEnd" est généré à la fin du déplacement du contrôle.

© 2000-2024, MetaQuotes Ltd.

DragObjectCreate

Crée un objet de déplacement

```
virtual bool  DragObjectCreate()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

DragObjectDestroy

Détruit un objet de déplacement

```
virtual bool  DragObjectDestroy()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CWndObj

CWndObj est une classe de base pour les contrôles simples (basés sur les objets graphiques) de la Bilibothèque Standard.

Description

La classe CWndObj implémente les méthodes de base des contrôles simples.

Déclaration

```
class CWndObj : public CWnd
```

Titre

```
#include <Controls\WndObj.mqh>
```

Hiérarchie d'héritage

COBject
  CWnd
    CWndObj

Descendants directs

CBmpButton, CButton, CEdit, CLabel, CPanel, CPicture

Méthodes de Classe

Traitement des évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
Texte	Retourne/Définit la propriété OBJPROP_TEXT de l'objet graphique
Color	Retourne/Définit la propriété OBJPROP_COLOR de l'objet graphique
ColorBackground	>Retourne/Définit la propriété OBJPROP_BGCOLOR de l'objet graphique
ColorBorder	>Retourne/Définit la propriété OBJPROP_BORDER_COLOR de l'objet graphique
Font	>Retourne/Définit la propriété OBJPROP_FONT de l'objet graphique
FontSize	>Retourne/Définit la propriété OBJPROP_FONTSIZE de l'objet graphique

© 2000-2024, MetaQuotes Ltd.

Traitement des évènements graphiques	
ZOrder	>Retourne/Définit la propriété OBJPROP_ZORDER de l'objet graphique
Gestionnaires d'évènements des objets graphiques	
OnObjectCreate	Gestionnaire d'évènement CHARTEVENT_OBJECT_CREATE
OnObjectChange	Gestionnaire d'évènement CHARTEVENT_OBJECT_CHANGE
OnObjectDelete	Gestionnaire d'évènement CHARTEVENT_OBJECT_DELETE
OnObjectDrag	Gestionnaire d'évènement CHARTEVENT_OBJECT_DRAGOpérations à la souris
Gestionnaires d'évènements de changement des propriétés	
OnSetText	Gestionnaire d'évènement "SetText"
OnSetColor	Gestionnaire d'évènement "SetColor"
OnSetColorBackground	Gestionnaire d'évènement "SetColorBackground"
OnSetFont	Gestionnaire d'évènement "SetFont"
OnSetFontSize	Gestionnaire d'évènement "SetFontSize"
OnSetZOrder	Gestionnaire d'évènement "SetZOrder"
Gestionnaires d'évènements internes	
OnDestroy	Gestionnaire d'évènement "Destroy"
OnChange	Gestionnaire d'évènement "Change"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CWnd

Create, Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

Text (Méthode "Get")

Retourne la propriété OBJPROP_TEXT (texte) de l'objet graphique.

```
string  Text()
```

Valeur de retour

La valeur de la propriété OBJPROP_TEXT.

Text (Méthode "Set")

Définit la propriété OBJPROP_TEXT (texte) de l'objet graphique.

```
bool  Text(
   const string  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_TEXT.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Color (Méthode "Get")

Retourne la propriété OBJPROP_COLOR (couleur) de l'objet graphique.

```
color  Color()
```

Valeur de retour

La valeur de la propriété OBJPROP_COLOR.

Color (Méthode "Set")

Définit la propriété OBJPROP_COLOR (couleur) de l'objet graphique.

```
bool  Color(
   const color  value      // valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_COLOR.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ColorBackground (Méthode "Get")

Retourne la valeur de la propriété OBJPROP_BGCOLOR (couleur d'arrière plan) de l'objet graphique.

```
color  ColorBackground()
```

Valeur de retour

La valeur de la propriété OBJPROP_BGCOLOR.

ColorBackground (Méthode "Set")

Définit la propriété OBJPROP_BGCOLOR (couleur d'arrière plan) de l'objet graphique.

```
bool  ColorBackground(
   const color  value      // valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_BGCOLOR.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ColorBorder (Méthode "Get")

Retourne la valeur de la propriété OBJPROP_BORDER_COLOR (couleur de la bordure) de l'objet graphique.

```
color  ColorBorder()
```

Valeur de retour

La valeur de la propriété OBJPROP_BORDER_COLOR.

ColorBorder (Méthode "Set")

Définit la valeur de la propriété OBJPROP_BORDER_COLOR (couleur de la bordure) de l'objet graphique.

```
bool  ColorBorder(
   const color  value      // valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_BORDER_COLOR.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Font (Méthode "Get")

Retourne la propriété OBJPROP_FONT (police de caractères) de l'objet graphique.

```
string  Font()
```

Valeur de retour

La valeur de la propriété OBJPROP_FONT.

Font (Méthode "Set")

Définit la propriété OBJPROP_FONT (police de caractères) de l'objet graphique.

```
bool  Font(
   const string  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_FONT.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

FontSize (Méthode "Get")

Retourne la propriété OBJPROP_FONTSIZE (taille de la police de caractères) de l'objet graphique.

```
int  FontSize()
```

Valeur de retour

La valeur de la propriété OBJPROP_FONTSIZE.

FontSize (Méthode "Set")

Définit la propriété OBJPROP_FONTSIZE (taille de la police de caractères) de l'objet graphique.

```
bool  FontSize(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_FONTSIZE.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ZOrder (Méthode "Get")

>Retourne/Définit la propriété OBJPROP_ZORDER de l'objet graphique

```
long  ZOrder()
```

Valeur de retour

La valeur de la propriété OBJPROP_ZORDER.

ZOrder (Méthode "Set")

>Définit la propriété OBJPROP_ZORDER de l'objet graphique

```
bool  ZOrder(
   const long  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_ZORDER.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnObjectCreate

Le gestionnaire d'évènement CHARTEVENT_OBJECT_CREATE de l'objet graphique.

```
virtual bool  OnObjectCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnObjectChange

Le gestionnaire d'évènement CHARTEVENT_OBJECT_CHANGE de l'objet graphique.

```
virtual bool  OnObjectChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnObjectDelete

Le gestionnaire d'évènement CHARTEVENT_OBJECT_DELETE de l'objet graphique.

```
virtual bool  OnObjectDelete()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnObjectDrag

Le gestionnaire d'évènement CHARTEVENT_OBJECT_DRAG de l'objet graphique.

```
virtual bool  OnObjectDrag()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetText

Le gestionnaire d'évènement "SetText" (changement de la propriété OBJPROP_TEXT) du contrôle.

```
virtual bool  OnSetText()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnSetColor

Le gestionnaire d'évènement "SetColor" (changement de la propriété OBJPROP_COLOR) du contrôle.

```
virtual bool  OnSetColor()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBackground

Le gestionnaire d'évènement "SetColorBackground" (changement de la propriété OBJPROP_BGCOLOR) du contrôle.

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnSetFont

Le gestionnaire d'évènement "SetFont" (changement de la propriété OBJPROP_FONT) du contrôle.

```
virtual bool  OnSetFont()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnSetFontSize

Le gestionnaire d'évènement "SetFontSize" (changement de la propriété OBJPROP_FONTSIZE) du contrôle.

```
virtual bool  OnSetFontSize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnSetZOrder

Le gestionnaire d'évènement "SetZOrder" (changement de la propriété OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnDestroy

Le gestionnaire d'évènement "Destroy" du contrôle.

```
virtual bool  OnDestroy()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CWndContainer

La classe CWndContainer est une classe de base pour les contrôles complexes (contenant des contrôles inter-dépendants) de la Bilbiothèque Standard MQL5 (MQL5 Standard Library).

Description

La classe CWndContainer implémente les méthodes de base des contrôles complexes.

Déclaration

```
class CWndContainer : public CWnd
```

Titre

```
#include <Controls\WndContainer.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndContainer

Descendants directs

CCheckBox, CComboBox, CDateDropList, CDatePicker, CDialog, CRadioButton, CScroll, CSpinEdit, CWndClient

Méthodes de Classe

Destruction	
Destroy	Détruit tous les contrôles du conteneur
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
OnMouseEvent	Le gestionnaire d'évènement CHARTEVENT_MOUSE_MOVE
Accès au conteneur	
ControlsTotal	Retourne le nombre total de contrôles dans le conteneur
Control	Retourne un contrôle par son index
ControlFind	Retourne un contrôle par son identifiant
Ajout/Suppression	
Add	Ajoute un contrôle dans le conteneur
Delete	Supprime un contrôle du conteneur

© 2000-2024, MetaQuotes Ltd.

Destruction	
Géometrie	
Move	Définit les nouvelles coordonnées de tous les contrôles du conteneur.
Shift	Effectue le décalage relatif des coordonnées de tous les contrôles du conteneur.
Identification	
Id	Définit l'identifiant de tous les contrôles du conteneur
Etat	
Enable	Active tous les contrôles du conteneur
Disable	Désactive tous les contrôles du conteneur
Show	Affiche tous les contrôles du conteneur
Hide	Cache tous les contrôles du conteneur
HandlerOpérations de la souris	
MouseFocusKill	Supprime le focus de la souris
Opérations avec des fichiers	
Save	Sauvegarde les informations du conteneur dans un fichier
Load	Charge les informations du conteneur depuis un fichier
Gestionnaires d'évènements internes	
OnResize	Gestionnaire d'évènement "Resize"
OnActivate	Gestionnaire d'évènement "Activate"
OnDeactivate	Gestionnaire d'évènement "Deactivate"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Create, Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2024, MetaQuotes Ltd.

Destroy

Détruit tous les contrôles du conteneur.

```
virtual bool  Destroy()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnMouseEvent

Gestionnaire d'évènements de la souris

```
virtual bool  OnMouseEvent(
   const int  x,          // coordonnée X
   const int  y,          // coordonnée Y
   const int  flags       // flags
   )
```

Paramètres

x

[in] Coordonnée X du curseur de la souris relativement au coin supérieur gauche du graphique.

y

[in] Coordonnée Y du curseur de la souris relativement au coin supérieur gauche du graphique.

flags

[in] Flag des états des boutons de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

ControlsTotal

Retourne le nombre total de contrôles du conteneur.

```
int ControlsTotal() const
```

Valeur de retour

Nombre de contrôles dans le conteneur.

© 2000-2024, MetaQuotes Ltd.

Control

Retourne un contrôle par son index.

```
CWnd*  Control(
   const int  ind      // index
   )  const
```

Paramètres

ind

[in] Index du contrôle.

Valeur de retour

Pointeur sur le contrôle, NULL si le contrôle n'est pas trouvé.

© 2000-2024, MetaQuotes Ltd.

ControlFind

Retourne un contrôle du conteneur par son identifiant.

```
virtual CWnd*  ControlFind(
   const long  id      // identifiant
   )
```

Paramètres

id

[in] Identificant du contrôle.

Valeur de retour

Pointeur sur le contrôle, NULL si le contrôle n'est pas trouvé.

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute un contrôle dans le conteneur.

```
bool  Add(
   CWnd&  control      // référence
   )
```

Paramètres

control

[in] Contrôle à ajouter, passé par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Delete

Supprime un contrôle du conteneur.

```
bool  Delete(
   CWnd&  control      // référence
   )
```

Paramètres

control

[in] Contrôle à supprimer, passé par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Move

Définit les nouvelles coordonnées de tous les contrôles du conteneur.

```
virtual bool  Move(
   const int  x,      // Coordonnée X
   const int  y       // Coordonnée Y
   )
```

Paramètres

x

[in] Nouvelle coordonnée X du coin supérieur gauche.

y

[in] Nouvelle coordonnée Y du coin supérieur gauche.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Shift

Effectue le décalage relatif des coordonnées de tous les contrôles du conteneur.

```
virtual bool  Shift(
   const int  dx,      // décalage x
   const int  dy       // décalage y
   )
```

Paramètres

dx

[in] Delta X.

dy

[in] Delta Y.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Id

Définit l'identifiant de tous les contrôles du conteneur.

```
virtual long  Id(
   const long  id      // identifiant
   )
```

Paramètres

id

[in] Identificant du groupe de base.

Valeur de retour

Nombre d'identifiants, utilisé par les contrôles du conteneur.

© 2000-2024, MetaQuotes Ltd.

Enable

Active tous les contrôles du conteneur.

```
virtual bool  Enable()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Disable

Désactive tous les contrôles du conteneur.

```
virtual bool  Disable()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Show

Affiche tous les contrôles du conteneur.

```
virtual bool  Show()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Hide

Cache tous les contrôles du conteneur.

```
virtual bool  Hide()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

MouseFocusKill

Réinitialise l'état sauvegardé des boutons de la souris et désactive tous les contrôles du conteneur.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // identifiant
   )
```

Paramètres

id=CONTROLS_INVALID_ID

[in] Identifiant du contrôle ayant reçu le focus de la souris.

Valeur de retour

Le résultat de la désactivation des contrôles.

© 2000-2024, MetaQuotes Ltd.

Save

Sauvegarde les informations du conteneur dans un fichier

```
virtual bool  Save(
   const int  file_handle      // handle
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire (doit être ouvert en écriture).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Load

Charge les informations du conteneur depuis un fichier

```
virtual bool  Load(
   const int  file_handle      // handle
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire (doit être ouvert en lecture).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnActivate

Le gestionnaire d'évènement "Activate" du contrôle.

```
virtual bool  OnActivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnDeactivate

Le gestionnaire d'évènement "Deactivate" du contrôle.

```
virtual bool  OnDeactivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

CLabel

La classe CLabel est une classe de contrôle simple, basé sur l'objet graphique "Etiquette de texte".

Description

La classe CLabel permet la création d'étiquettes de texte simples.

Déclaration

```
class CLabel : public CWndObj
```

Titre

```
#include <Controls\Label.mqh>
```

Hiérarchie d'héritage

COBJECT
CWnd
CWndObj
CLabel

Résultat du code fournit ci-dessous :

Méthodes de Classe

Create	
Création	Crée le contrôle

© 2000-2024, MetaQuotes Ltd.

Create	
Gestionnaires d'évènements de changement des propriétés	
OnSetText	Gestionnaire d'évènement "SetText"
OnSetColor	Gestionnaire d'évènement "SetColor"
OnSetFont	Gestionnaire d'évènement "SetFont"
OnSetFontSize	Gestionnaire d'évènement "SetFontSize"
Gestionnaires d'évènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemple de création d'un panneau avec une étiquette de texte :

```
//+------------------------------------------------------------------+
//|                                                ControlsLabel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CLab
#include <Controls\Dialog.mqh>
#include <Controls\Label.mqh>
```

```
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CLabel            m_label;                          // Objet CLabel
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
protected:
   //--- crée des contrôles dépendants
   bool              CreateLabel(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnClickLabel(void);
  };
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)

EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
```

```
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateLabel())
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée le "CLabel"                                                 |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateLabel(void)
  {
//--- coordonnées
   int x1=INDENT_RIGHT;
   int y1=INDENT_TOP+CONTROLS_GAP_Y;
   int x2=x1+100;
   int y2=y1+20;
//--- création
   if(!m_label.Create(m_chart_id,m_name+"Label",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_label.Text("Label"))
      return(false);
   if(!Add(m_label))
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert                            |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre d'évènement de type long
                  const double& dparam, // paramètre d'évènement de type double
                  const string& sparam) // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CLabel.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnSetText

Le gestionnaire d'évènement "SetText" (changement de la propriété OBJPROP_TEXT) du contrôle.

```
virtual bool  OnSetText()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetColor

Le gestionnaire d'évènement "SetColor" (changement de la propriété OBJPROP_COLOR) du contrôle.

```
virtual bool  OnSetColor()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetFont

Le gestionnaire d'évènement "SetFont" (changement de la propriété OBJPROP_FONT) du contrôle.

```
virtual bool  OnSetFont()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetFontSize

Le gestionnaire d'évènement "SetFontSize" (changement de la propriété OBJPROP_FONTSIZE) du contrôle.

```
virtual bool  OnSetFontSize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CBmpButton

La classe CBmpButton est une classe de contrôle simple, basé sur l'objet graphique "Etiquette Bitmap".

Description

La classe CBmpButton permet la création de boutons avec une image.

Déclaration

```
class CBmpButton : public CWndObj
```

Titre

```
#include <Controls\BmpButton.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndObj
      CBmpButton

Résultat du code fournit ci-dessous :

Méthodes de Classe

Create	
Create	Crée le contrôle

© 2000-2024, MetaQuotes Ltd.

Create	
Propriétés	
Border	Retourne/Définit la propriété "Border" du contrôle
BmpNames	Définit le nom des fichiers bmp du contrôle
BmpOffName	Retourne/Définit le nom du fichier bmp pour l'état Eteint (Off)
BmpOnName	Retourne/Définit le nom du fichier bmp pour l'état Allumé (On)
BmpPassiveName	Retourne/Définit le nom du fichier bmp pour l'état passif
BmpActiveName	Retourne/Définit le nom du fichier bmp pour l'état actif
Etat	
Pressed	Retourne/Définit l'état du contrôle
Locking	Retourne/Définit la propriété "Locking" du contrôle
Gestionnaires d'évènements internes	
OnSetZOrder	Gestionnaire d'évènement "SetZOrder"
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"
OnChange	Gestionnaire d'évènement "Change"
OnActivate	Gestionnaire d'évènement "Activate"
OnDeactivate	Gestionnaire d'évènement "Deactivate"
OnMouseDown	Gestionnaire d'évènement "MouseDown"
OnMouseUp	Gestionnaire d'évènement "MouseUp"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndObj

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemple de création d'un panneau avec une Etiquette Bitmap :

```
//+------------------------------------------------------------------+
//|                                              ControlsBmpButton.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CBmp
#include <Controls\Dialog.mqh>
#include <Controls\BmpButton.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CBmpButton        m_bmpbutton1;                    // Objet CBmpButton
   CBmpButton        m_bmpbutton2;                    // Objet CBmpButton
```

```
public:
                      CControlsDialog(void);
                     ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crée des contrôles dépendants
   bool              CreateBmpButton1(void);
   bool              CreateBmpButton2(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnClickBmpButton1(void);
   void              OnClickBmpButton2(void);
  };
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_bmpbutton1,OnClickBmpButton1)
ON_EVENT(ON_CLICK,m_bmpbutton2,OnClickBmpButton2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateBmpButton1())
      return(false);
   if(!CreateBmpButton2())
      return(false);
//--- succès
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Crée le bouton "BmpButton1"                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton1(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- création
   if(!m_bmpbutton1.Create(m_chart_id,m_name+"BmpButton1",m_subwin,x1,y1,x2,y2))
      return(false);
//--- définit le nom des fichiers bmp du contrôle CBmpButton
   m_bmpbutton1.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton1))
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée le bouton fixe "BmpButton2"                                 |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton2(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- création
   if(!m_bmpbutton2.Create(m_chart_id,m_name+"BmpButton2",m_subwin,x1,y1,x2,y2))
      return(false);
//--- définit le nom des fichiers bmp du contrôle CBmpButton
   m_bmpbutton2.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton2))
      return(false);
   m_bmpbutton2.Locking(true);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton1(void)
  {
   Comment(__FUNCTION__);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton2(void)
  {
   if(m_bmpbutton2.Pressed())
      Comment(__FUNCTION__+" L'état du contrôle est : On");
   else
      Comment(__FUNCTION__+" L'état du contrôle est : Off");
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert       |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre d'évènement de type long
                  const double& dparam, // paramètre d'évènement de type double
                  const string& sparam) // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CBmpButton.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Border (Méthode "Get")

Retourne la propriété "Border" (épaisseur de la bordure) du contrôle.

```
int  Border()  const
```

Valeur de retour

La propriété "Border".

Border (Méthode "Set")

Définit la propriété "Border" (épaisseur de la bordure) du contrôle.

```
bool  Border(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "Border".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

BmpNames

Définit le nom des fichiers bmp du contrôle

```
bool  BmpNames(
   const string  off="",      // nom du fichier
   const string  on=""        // nom du fichier
   )
```

Paramètres

off=""

[in] Nom du fichier bmp pour l'état Eteint (OFF).

on=""

[in] Nom du fichier bmp pour l'état Allumé (ON).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

BmpOffName (Méthode "Get")

Retourne le nom du fichier bmp pour l"état Eteint (OFF).

```
string  BmpOffName()  const
```

Valeur de retour

Nom du fichier bmp pour l'état Eteint (OFF).

BmpOffName (Méthode "Set")

Définit le nom du fichier bmp pour l'état Eteint (OFF).

```
bool  BmpOffName(
   const string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nom du fichier bmp pour l'état Eteint (OFF).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

BmpOnName (Méthode "Get")

Retourne le nom du fichier bmp pour l'état Allumé (ON).

```
string  BmpOnName()  const
```

Valeur de retour

Nom du fichier bmp pour l'état Allumé (ON).

BmpOnName (Méthode "Set")

Définit le nom du fichier bmp pour l'état Allumé (ON).

```
bool  BmpOnName(
   const string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nom du fichier bmp pour l'état Allumé (ON).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

BmpPassiveName (Méthode "Get")

Retourne le nom du fichier bmp pour l'état passif du contrôle.

```
string  BmpPassiveName()  const
```

Valeur de retour

Nom du fichier bmp pour l'état passif du contrôle.

BmpPassiveName (Méthode "Set")

Définit le nom du fichier bmp pour l'état passif.

```
bool  BmpPassiveName(
   const string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nom du fichier bmp pour l'état passif du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

BmpActiveName (Méthode "Get")

Retourne le nom du fichier bmp pour l'état actif.

```
string  BmpActiveName()  const
```

Valeur de retour

Nom du fichier bmp pour l'état actif.

Note

Le contrôle devient actif lorsque le curseur de la souris passe au-dessus.

BmpActiveName (Méthode "Set")

Définit le nom du fichier bmp pour l'état actif

```
bool  BmpActiveName(
   const string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nom du fichier bmp pour l'état actif.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Pressed (Méthode "Get")

Retourne l'état (propriété "Pressed") du contrôle.

```
bool  Pressed()  const
```

Valeur de retour

Etat du contrôle.

Pressed (Méthode "Set")

Définit l'état (propriété "Pressed") du contrôle.

```
bool  Pressed(
   const bool  pressed      // nouvel état
   )
```

Paramètres

pressed

[in] Nouvel état du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Locking (Méthode "Get")

Retourne la propriété "Locking" du contrôle.

```
bool  Locking()  const
```

Valeur de retour

Valeur de la propriété "Locking".

Locking (Méthode "Set")

Définit la nouvelle valeur de la propriété "Locking" du contrôle.

```
void  Locking(
   const bool  locking      // nouvelle valeur
   )
```

Paramètres

locking

[in] Nouvelle valeur pour la propriété "Locking".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnSetZOrder

Le gestionnaire d'évènement "SetZOrder" (changement de la propriété OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnActivate

Le gestionnaire d'évènement "Activate" du contrôle.

```
virtual bool  OnActivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnDeactivate

Le gestionnaire d'évènement "Deactivate" du contrôle.

```
virtual bool  OnDeactivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnMouseDown

Le gestionnaire d'évènement "MouseDown" du contrôle.

```
virtual bool  OnMouseDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseDown" est généré lorsque le bouton gauche de la souris est appuyé sur le contrôle.

© 2000-2024, MetaQuotes Ltd.

OnMouseUp

Le gestionnaire d'évènement "MouseUp" du contrôle (le bouton gauche de la souris est relâché).

```
virtual bool  OnMouseUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseUp" est généré lorsque le bouton gauche de la souris est relâché au-dessus du contrôle.

© 2000-2024, MetaQuotes Ltd.

CButton

La classe CButton est une classe de contrôle simple, basé sur l'objet graphique "Bouton".

Description

La classe CButton est une classe pour la création de boutons simples.

Déclaration

```
class CButton : public CWndObj
```

Titre

```
#include <Controls\Button.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndObj
      CButton

Résultat du code fournit ci-dessous :

Méthodes de Classe

Création	
Create	Crée le contrôle

© 2000-2024, MetaQuotes Ltd.

Création	
Etat	
Pressed	Retourne/Définit la propriété "Pressed"
Locking	Retourne/Définit la propriété "Locking"
Gestionnaires d'évènements de changement des propriétés	
OnSetText	Gestionnaire d'évènement "SetText"
OnSetColor	Gestionnaire d'évènement "SetColor"
OnSetColorBackground	Gestionnaire d'évènement "SetColorBackground"
OnSetColorBorder	Gestionnaire d'évènement "SetColorBorder"
OnSetFont	Gestionnaire d'évènement "SetFont"
OnSetFontSize	Gestionnaire d'évènement "SetFontSize"
Gestionnaires d'évènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"
OnResize	Gestionnaire d'évènement "Resize"
OnMouseDown	Gestionnaire d'évènement "MouseDown"
OnOnMouseUp	Gestionnaire d'évènement "MouseUp"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

© 2000-2024, MetaQuotes Ltd.

Exemple de création d'un panneau avec un bouton :

```
//+------------------------------------------------------------------+
//|                                                ControlsButton.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CBut
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CButton           m_button1;                       // Objet bouton
   CButton           m_button2;                       // Objet bouton
   CButton           m_button3;                       // Objet bouton fixe

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
```

```
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crée des contrôles dépendants
   bool              CreateButton1(void);
   bool              CreateButton2(void);
   bool              CreateButton3(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnClickButton1(void);
   void              OnClickButton2(void);
   void              OnClickButton3(void);
  };
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_button1,OnClickButton1)
ON_EVENT(ON_CLICK,m_button2,OnClickButton2)
ON_EVENT(ON_CLICK,m_button3,OnClickButton3)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateButton1())
      return(false);
   if(!CreateButton2())
      return(false);
   if(!CreateButton3())
      return(false);
//--- succès
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Crée le bouton "Button1"                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton1(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- création
   if(!m_button1.Create(m_chart_id,m_name+"Button1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button1.Text("Button1"))
      return(false);
   if(!Add(m_button1))
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée le bouton "Button2"                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton2(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT+(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- création
   if(!m_button2.Create(m_chart_id,m_name+"Button2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button2.Text("Button2"))
      return(false);
   if(!Add(m_button2))
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée le bouton fixe "Button3"                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton3(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
```

© 2000-2024, MetaQuotes Ltd.

```
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- création
   if(!m_button3.Create(m_chart_id,m_name+"Button3",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button3.Text("Locked"))
      return(false);
   if(!Add(m_button3))
      return(false);
   m_button3.Locking(true);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton1(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton2(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton3(void)
  {
   if(m_button3.Pressed())
      Comment(__FUNCTION__+" Etat du contrôle : On");
   else
      Comment(__FUNCTION__+" Etat du contrôle : Off");
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert       |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprime le commentaire
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre d'évènement de type long
                  const double& dparam, // paramètre d'évènement de type double
                  const string& sparam) // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CButton.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Pressed (Méthode "Get")

Retourne l'état (propriété "Pressed") du contrôle.

```
bool  Pressed()  const
```

Valeur de retour

Etat du contrôle.

Pressed (Méthode "Set")

Définit l'état (propriété "Pressed") du contrôle.

```
bool  Pressed(
   const bool  pressed      // nouvel état
   )
```

Paramètres

pressed

[in] Nouvel état du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Locking (Méthode "Get")

Retourne la propriété "Locking" du contrôle.

```
bool  Locking()  const
```

Valeur de retour

Valeur de la propriété "Locking".

Locking (Méthode "Set")

Définit la nouvelle valeur de la propriété "Locking" du contrôle.

```
void  Locking(
   const bool  locking      // nouvelle valeur
   )
```

Paramètres

locking

[in] Nouvelle valeur pour la propriété "Locking".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnSetText

Le gestionnaire d'évènement "SetText" (changement de la propriété OBJPROP_TEXT) du contrôle.

```
virtual bool  OnSetText()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetColor

Le gestionnaire d'évènement "SetColor" (changement de la propriété OBJPROP_COLOR) du contrôle.

```
virtual bool  OnSetColor()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBackground

Le gestionnaire d'évènement "SetColorBackground" (changement de la propriété OBJPROP_BGCOLOR) du contrôle.

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBorder

Le gestionnaire d'évènement "SetColorBorder" (changement de la propriété OBJPROP_BORDER_COLOR) du contrôle.

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetFont

Le gestionnaire d'évènement "SetFont" (changement de la propriété OBJPROP_FONT) du contrôle.

```
virtual bool  OnSetFont()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetFontSize

Le gestionnaire d'évènement "SetFontSize" (changement de la propriété OBJPROP_FONTSIZE) du contrôle.

```
virtual bool  OnSetFontSize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnMouseDown

Le gestionnaire d'évènement "MouseDown" du contrôle.

```
virtual bool  OnMouseDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseDown" est généré lorsque le bouton gauche de la souris est appuyé sur le contrôle.

© 2000-2024, MetaQuotes Ltd.

OnMouseUp

Le gestionnaire d'évènement "MouseUp" du contrôle (le bouton gauche de la souris est relâché).

```
virtual bool  OnMouseUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseUp" est généré lorsque le bouton gauche de la souris est relâché au-dessus du contrôle.

© 2000-2024, MetaQuotes Ltd.

CEdit

La classe CEdit est une classe de contrôle simple, basé sur l'objet graphique "Edit".

Description

La classe CButton est une classe pour la création de contrôles dans lesquels l'utilisateur peut saisir un texte.

Déclaration

```
class CEdit : public CWndObj
```

Titre

```
#include <Controls\Edit.mqh>
```

Hiérarchie d'héritage

COBJECT
  CWnd
    CWndObj
      CEdit

Résultat du code fournit ci-dessous :

Méthodes de Classe

Création	
Create	Crée le contrôle

© 2000-2024, MetaQuotes Ltd.

Création	
Propriétés	
ReadOnly	Retourne/Définit la propriété "ReadOnly"
TextAlign	Retourne/Definit la propriété "TextAlign"
Gestionnaires d'évènements des objets graphiques	
OnObjectEndEdit	Gestionnaire (virtuel) de l'évènement CHARTEVENT_OBJECT_ENDEDIT
Gestionnaires d'évènements de changement des propriétés	
OnSetText	Gestionnaire d'évènement "SetText"
OnSetColor	Gestionnaire d'évènement "SetColor"
OnSetColorBackground	Gestionnaire d'évènement "SetColorBackground"
OnSetColorBorder	Gestionnaire d'évènement "SetColorBorder"
OnSetFont	Gestionnaire d'évènement "SetFont"
OnSetFontSize	Gestionnaire d'évènement "SetFontSize"
OnSetZOrder	Gestionnaire d'évènement "SetZOrder"
Gestionnaires d'évènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"
OnResize	Gestionnaire d'évènement "Resize"
OnChange	Gestionnaire d'évènement "Change"
OnClick	Gestionnaire d'évènement "Click"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndObj

Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemple de création d'un panneau avec un contrôle Zone d'Edition :

```
//+------------------------------------------------------------------+
//|                                                ControlsEdit.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CEdi
#include <Controls\Dialog.mqh>
#include <Controls\Edit.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
```

```
private:
   CEdit             m_edit;                          // Objet CEdit

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique

protected:
   //--- crée des contrôles dépendants
   bool              CreateEdit(void);
  };
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateEdit())
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée le champ d'affichage                                        |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateEdit(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=ClientAreaWidth()-INDENT_RIGHT;
   int y2=y1+EDIT_HEIGHT;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- création
   if(!m_edit.Create(m_chart_id,m_name+"Edit",m_subwin,x1,y1,x2,y2))
      return(false);
//--- permet l'édition du contenu
   if(!m_edit.ReadOnly(false))
      return(false);
   if(!Add(m_edit))
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert       |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprime le commentaire
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // identifiant de l'évènement
                  const long& lparam,    // paramètre d'évènement de type long
                  const double& dparam,  // paramètre d'évènement de type double
                  const string& sparam)  // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CEdit.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ReadOnly (Méthode "Get")

Retourne la propriété "ReadOnly" du contrôle.

```
bool  ReadOnly()
```

Valeur de retour

La valeur de la propriété "ReadOnly".

ReadOnly (Méthode "Set")

Définit la valeur de la propriété "ReadOnly" du contrôle.

```
bool  ReadOnly(
   const bool  flag      // nouvelle valeur
   )
```

Paramètres

flag

[in] Nouvelle valeur de la propriété "ReadOnly".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

TextAlign (Méthode "Get")

Retourne la valeur de la propriété "TextAlign" (mode d'alignement du texte).

```
ENUM_ALIGN_MODE  TextAlign() const
```

Valeur de retour

Définit la valeur de la propriété "TextAlign" du contrôle.

TextAlign (Méthode "Set")

Définit la nouvelle valeur de la propriété "TextAlign" (mode d'alignement du texte) du contrôle.

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // nouvelle valeur
   )
```

Paramètres

align

[in] Nouvelle valeur de la propriété "TextAlign".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2024, MetaQuotes Ltd.

OnObjectEndEdit

Gestionnaire de l'évènement CHARTEVENT_OBJECT_ENDEDIT

```
virtual bool  OnObjectEndEdit()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetText

Le gestionnaire d'évènement "SetText" (changement de la propriété OBJPROP_TEXT) du contrôle.

```
virtual bool  OnSetText()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetColor

Le gestionnaire d'évènement "SetColor" (changement de la propriété OBJPROP_COLOR) du contrôle.

```
virtual bool  OnSetColor()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBackground

Le gestionnaire d'évènement "SetColorBackground" (changement de la propriété OBJPROP_BGCOLOR) du contrôle.

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBorder

Le gestionnaire d'évènement "SetColorBorder" (changement de la propriété OBJPROP_BORDER_COLOR<t3>) du contrôle.</t3>

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetFont

Le gestionnaire d'évènement "SetFont" (changement de la propriété OBJPROP_FONT) du contrôle.

```
virtual bool  OnSetFont()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetFontSize

Le gestionnaire d'évènement "SetFontSize" (changement de la propriété OBJPROP_FONTSIZE) du contrôle.

```
virtual bool  OnSetFontSize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetZOrder

Le gestionnaire d'évènement "SetZOrder" (changement de la propriété OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnClick

Le gestionnaire d'évènement "Click" (clic avec le bouton gauche de la souris) du contrôle.

```
virtual bool  OnClick()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CPanel

La classe CPanel est une classe de contrôle simple, basé sur l'objet graphique "Etiquette rectangulaire".

Description

La classe CPanel permet de combiner des contrôles ayant des fonctions similaires dans le groupe.

Déclaration

```
class CPanel : public CWndObj
```

Titre

```
#include <Controls\Panel.mqh>
```

Hiérarchie d'héritage

COBJECT
CWnd
CWndObj
CPanel

Résultat du code fournit ci-dessous :

Méthodes de Classe

Création	
Create	Crée le contrôle

© 2000-2024, MetaQuotes Ltd.

Création	
Propriétés de l'objet graphique	
BorderType	Retourne la propriété "BorderType" de l'objet graphique
Gestionnaires d'évènements des objets graphiques	
OnSetText	Gestionnaire d'évènement "SetText"
OnSetColorBackground	Gestionnaire d'évènement "SetColorBackground"
OnSetColorBorder	Gestionnaire d'évènement "SetColorBorder"
Gestionnaires d'évènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"
OnResize	Gestionnaire d'évènement "Resize"
OnChange	Gestionnaire d'évènement "Change"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemple de création d'un panneau avec une étiquette Rectangle :

```
//+------------------------------------------------------------------+
//|                                                ControlsPanel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
```

```
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CPar
#include <Controls\Dialog.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const

protected:
   //--- crée des contrôles dépendants
   bool              CreatePanel(void);
  };
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreatePanel())
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée le "CPanel"                                                 |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePanel(void)
  {
//--- coordonnées
   int x1=20;
   int y1=20;
   int x2=ExtDialog.Width()/3;
   int y2=ExtDialog.Height()/3;
//--- création
   if(!my_white_border.Create(0,ExtDialog.Name()+"MyWhiteBorder",m_subwin,x1,y1,x2,y2)
      return(false);
   if(!my_white_border.ColorBackground(CONTROLS_DIALOG_COLOR_BG))
      return(false);
   if(!my_white_border.ColorBorder(CONTROLS_DIALOG_COLOR_BORDER_LIGHT))
      return(false);
   if(!ExtDialog.Add(my_white_border))
      return(false);
   my_white_border.Alignment(WND_ALIGN_CLIENT,0,0,0,0);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//---
CPanel   my_white_border;        // Objet CPanel
bool     pause=true;             // true - pause
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert                            |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int OnInit()
  {
//---
   EventSetTimer(3);
   pause=true;
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprime le commentaire
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre d'évènement de type long
                  const double& dparam, // paramètre d'évènement de type double
                  const string& sparam) // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
//+------------------------------------------------------------------+
//|  Timer                                                           |
//+------------------------------------------------------------------+
void OnTimer()
  {
   pause=!pause;
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CPanel.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

BorderType (Méthode "Get")

Retourne la propriété "BorderType" de l'objet graphique.

```
ENUM_BORDER_TYPE  BorderType()
```

Valeur de retour

La valeur de la propriété "BorderType".

BorderType (Méthode "Set")

Définit la nouvelle valeur de la propriété "BorderType" de l'objet graphique.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // valeur
   )
```

Paramètres

type

[in] Nouvelle valeur de la propriété "BorderType".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnSetText

Le gestionnaire d'évènement "SetText" (changement de la propriété OBJPROP_TEXT) du contrôle.

```
virtual bool  OnSetText()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBackground

Le gestionnaire d'évènement "SetColorBackground" (changement de la propriété OBJPROP_BGCOLOR) du contrôle.

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBorder

Le gestionnaire d'évènement "SetColorBorder" (changement de la propriété OBJPROP_BORDER_COLOR<t3>) du contrôle.</t3>

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CPicture

La classe CPicture est une classe de contrôle simple, basé sur l'objet graphique "Etiquette Bitmap".

Description

La classe CPicture permet la création d'images graphiques simples.

Déclaration

```
class CPicture : public CWndObj
```

Titre

```
#include <Controls\Picture.mqh>
```

Hiérarchie d'héritage

COBJECT
CWnd
CWndObj
CPicture

Résultat du code fournit ci-dessous :

Méthodes de Classe

Création	
Create	Crée le contrôle

© 2000-2024, MetaQuotes Ltd.

Création	
Propriétés de l'objet graphique	
Border	Retourne/Définit l'épaisseur du contour de l'objet graphique
BmpName	Retourne/Définit le nom du fichier bmp du contrôle
Evènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"
OnChange	Gestionnaire d'évènement "Change"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Méthodes héritées de la classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemple de création d'un panneau avec une Etiquette Bitmap :

```
//+------------------------------------------------------------------+
//|                                                ControlsPicture.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CPic
#include <Controls\Dialog.mqh>
#include <Controls\Picture.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
```

```
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CPicture          m_picture;                       // Objet CPicture

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crée des contrôles dépendants
   bool              CreatePicture(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnClickPicture(void);
  };
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_picture,OnClickPicture)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreatePicture())
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée le "Picture"                                                |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePicture(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+32;
   int y2=y1+32;
//--- création
   if(!m_picture.Create(m_chart_id,m_name+"Picture",m_subwin,x1,y1,x2,y2))
      return(false);
//--- définit le nom des fichiers bmp pour afficher le contrôle CPicture
   m_picture.BmpName("\\Images\\euro.bmp");

   if(!Add(m_picture))
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickPicture(void)
  {
   Comment(__FUNCTION__);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert       |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprime le commentaire
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre d'évènement de type long
                  const double& dparam, // paramètre d'évènement de type double
                  const string& sparam) // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CPicture.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Border (Méthode "Get")

Retourne la propriété "Border" (épaisseur de la bordure) du contrôle.

```
int  Border()  const
```

Valeur de retour

La propriété "Border".

Border (Méthode "Set")

Définit la propriété "Border" (épaisseur de la bordure) du contrôle.

```
bool  Border(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "Border".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

BmpName (Méthode "Get")

Retourne le nom du fichier bmp du contrôle.

```
string  BmpName()  const
```

Valeur de retour

Nom du fichier bmp du contrôle.

BmpName (Méthode "Set")

Définit le nom du fichier bmp du contrôle.

```
bool  BmpName(
   const string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nom du fichier bmp du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CScroll

La classe CCanvas est la classe de base pour créer des barres de défilement.

Description

CScroll est un contrôle complexe (contenant différents contrôles), contenant la fonctionnalité de base de création des barres de défilement. La classe de base elle-même n'est pas utilisée comme un contrôle séparé, mais deux de ses descendants (classes CScrollV et CScrollH) le sont.

Déclaration

```
class CScroll : public CWndContainer
```

Titre

```
#include <Controls\Scrolls.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndContainer
      CScroll

Descendants directs

CScrollH, CScrollV

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements des objets graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
MinPos	Retourne/Définit la position minimale
MaxPos	Retourne/Définit la position maximale
CurrPos	Retourne/Définit la position courante
Création des contrôles dépendants	
CreateBack	Crée le bouton d'arrière plan
CreateInc	Crée le bouton d'incrémentation de la barre de défilement

© 2000-2024, MetaQuotes Ltd.

Création	
CreateDec	Crée le bouton de décrémentation de la barre de défilement
CreateThumb	Crée l'ascenseur (pouvant être déplacé) de la barre de défilement
Gestionnaires d'évènements des contrôles dépendants	
OnClickInc	Gestionnaire d'évènement utilisé pour gérer les évènements du bouton d'incrémentation
OnClickDec	Gestionnaire d'évènement utilisé pour gérer les évènements du boutonu de décrémentation
Gestionnaires d'évènements internes	
OnShow	Gestionnaire d'évènement "Create"
OnHide	Gestionnaire d'évènement "Hide"
OnChangePos	Gestionnaire d'évènement "ChangePosition"
Gestionnaires de déplacement de l'ascenseur	
OnThumbDragStart	Gestionnaire d'évènement "ThumbDragStart"
OnThumbDragProcess	Gestionnaire d'évènement "ThumbDragProcess"
OnThumbDragEnd	Gestionnaire d'évènement "ThumbDragEnd"
Position	
CalcPos	Retourne la position de la barre de défilement par ses coordonnées

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CScroll.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

MinPos (Méthode "Get")

Retourne la valeur de la propriété "MinPos" (position minimale) du contrôle CScroll.

```
int  MinPos()  const
```

Valeur de retour

Nouvelle valeur de la propriété "MinPos".

MinPos (Méthode "Set")

Définit la valeur de la propriété "MinPos" (position minimale) du contrôle CScroll.

```
void  MinPos(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "MinPos".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

MaxPos (Méthode "Get")

Définit la valeur de la propriété "MaxPos" (position maximale) du contrôle CScroll.

```
int  MaxPos()  const
```

Valeur de retour

Nouvelle valeur de la propriété "MaxPos".

MaxPos (Méthode "Set")

Définit la valeur de la propriété "MaxPos" (position maximale) du contrôle CScroll.

```
void  MaxPos(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriéé "MaxPos".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CurrPos (Méthode "Get")

Retourne la valeur de la propriété "CurrPos" (position actuelle) du contrôle CScroll.

```
int  CurrPos()  const
```

Valeur de retour

[in] Nouvelle valeur de la propriété "CurrPos".

CurrPos (Méthode "Set")

Définit la valeur de la propriété "CurrPos" (position actuelle) du contrôle CScroll.

```
void  CurrPos(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "Caption".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CreateBack

Crée le bouton d'arrière plan du contrôle CScroll.

```
virtual bool  CreateBack()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateInc

Crée le bouton d'incrémentation du contrôle CScroll.

```
virtual bool  CreateInc()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateDec

Crée le bouton de décrémentation du contrôle CScroll.

```
virtual bool  CreateDec()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateThumb

Crée l'ascenseur (pouvant être déplacé) du contrôle CScroll

```
virtual bool  CreateThumb()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnClickInc

Le gestionnaire d'évènement "ClickInc" (clic avec le bouton gauche de la souris sur le bouton d'incrémentation) du contrôle.

```
virtual bool  OnClickInc()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnClickDec

Le gestionnaire d'évènement "ClickDec" (clic avec le bouton gauche de la souris sur le bouton de décrémentation) du contrôle.

```
virtual bool  OnClickDec()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChangePos

Le gestionnaire d'évènement "ChangePos" (changement de position) du contrôle.

```
virtual bool  OnChangePos()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragStart

Le gestionnaire d'évènement "ThumbDragStart" (début du déplacement de l'ascenceur) du contrôle.

```
virtual bool  OnThumbDragStart()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragStart" est généré au démarrage du déplacement de l'ascenseur.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragProcess

Le gestionnaire d'évènement "ThumbDragProcess" du contrôle.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // Coordonnée X
   const int  y       // Coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X actuelle du curseur de la souris.

y

[in] Coordonnée Y actuelle du curseur de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragProcess" est généré lorsque le contrôle de la barre de défilement (l'ascenseur) est déplacé.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragEnd

Le gestionnaire d'évènement "ThumbDragEnd" (fin du déplacement de l'ascenseur) du contrôle.

```
virtual bool  OnThumbDragEnd()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragEnd" est généré lorsque le déplacement du contrôle de la barre de défilement (l'ascenseur) est terminé.

© 2000-2024, MetaQuotes Ltd.

CalcPos

Retourne la position de la barre de défilement par ses coordonnées

```
virtual int  CalcPos(
   const int      coord      // coordonnée
   )
```

Paramètres

coord

[in] Coordonnée de la barre de défilement.

Valeur de retour

Position de la barre de défilement.

© 2000-2024, MetaQuotes Ltd.

CScrollV

La classe CScrollV est la classe du contrôle complexe "Barre de défilement verticale".

Description

La classe CScrollV permet la création des barres de défilement verticale.

Déclaration

```
class CScrollV : public CScroll
```

Titre

```
#include <Controls\Scrolls.mqh>
```

Hiérarchie d'héritage

- CObject
  - CWnd
    - CWndContainer
      - CScroll
        - CScrollV

Résultat du code fournit ci-dessous :

Méthodes de Classe

© 2000-2024, MetaQuotes Ltd.

Contrôles dépendants	
CreateInc	Cré le bouton d'incrémentation de la barre de défilement
CreateDec	Cré le bouton de décrémentation de la barre de défilement
CreateThumb	Crée le bouton de l'ascenseur (pouvant être déplacé)
Gestionnaires d'évènements internes	
OnResize	Gestionnaire d'évènement "Resize"
OnChangePos	Gestionnaire d'évènement "ChangePosition"
Gestionnaires d'évènement de Drag n' Drop	
OnThumbDragStart	Gestionnaire d'évènement "ThumbDragStart"
OnThumbDragProcess	Gestionnaire d'évènement "ThumbDragProcess"
OnThumbDragEnd	Gestionnaire d'évènement "ThumbDragEnd"
Position	
CalcPos	Retourne la position de la barre de défilement par ses coordonnées

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Méthodes héritées de la classe CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Exemple de création d'un panneau avec une barre de défilement verticale :

```
//+------------------------------------------------------------------+
//|                                              ControlsScrollV.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
```

```
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CSc
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollV          m_scroll_v;                     // Objet CScrollV

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crée des contrôles dépendants
   bool              CreateScrollV(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
```

```
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateScrollV())
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée l'objet CScrollsV                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollV(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+18;
   int y2=y1+LIST_HEIGHT;
//--- création
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollV",m_subwin,x1,y1,x2,y2))
      return(false);
//--- définit la barre de défilement
   m_scroll_v.MinPos(0);
//--- définit la barre de défilement
   m_scroll_v.MaxPos(10);
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!Add(m_scroll_v))
      return(false);
   Comment("Position de la barre de défilement ",m_scroll_v.CurrPos());
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position de la barre de défilement ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position de la barre de défilement ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert       |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                       |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprime le commentaire
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // identifiant de l'évènement
                  const long& lparam,    // paramètre d'évènement de type long
                  const double& dparam,  // paramètre d'évènement de type double
                  const string& sparam)  // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

CreateInc

Crée le bouton d'incrémentation du contrôle.

```
virtual bool  CreateInc()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateDec

Crée le bouton de décrémentation du contrôle.

```
virtual bool  CreateDec()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateThumb

Crée l'ascenseur (pouvant être déplacé) du contrôle.

```
virtual bool  CreateThumb()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChangePos

Le gestionnaire d'évènement "ChangePos" (changement de position) du contrôle.

```
virtual bool  OnChangePos()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragStart

Le gestionnaire d'évènement "ThumbDragStart" (début du déplacement de l'ascenceur) du contrôle.

```
virtual bool  OnThumbDragStart()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragStart" est généré au démarrage du déplacement de l'ascenseur.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragProcess

Le gestionnaire d'évènement "ThumbDragProcess" du contrôle.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // Coordonnée X
   const int  y       // Coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X actuelle du curseur de la souris.

y

[in] Coordonnée Y actuelle du curseur de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragProcess" est généré lorsque le contrôle de la barre de défilement (l'ascenseur) est déplacé.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragEnd

Le gestionnaire d'évènement "ThumbDragEnd" (fin du déplacement de l'ascenseur) du contrôle.

```
virtual bool  OnThumbDragEnd()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragEnd" est généré lorsque le déplacement du contrôle de la barre de défilement (l'ascenseur) est terminé.

© 2000-2024, MetaQuotes Ltd.

CalcPos

Retourne la position de la barre de défilement par ses coordonnées

```
virtual int  CalcPos(
   const int      coord      // coordonnée
   )
```

Paramètres

coord

[in] Coordonnée de la barre de défilement.

Valeur de retour

Position de la barre de défilement.

© 2000-2024, MetaQuotes Ltd.

CScrollH

La classe CScrollH est la classe du contrôle complexe "Barre de défilement horizontale".

Description

La classe CScrollH permet la création des barres de défilement horizontale.

Déclaration

```
class CScrollH : public CScroll
```

Titre

```
#include <Controls\Scrolls.mqh>
```

Hiérarchie d'héritage

- CObject
  - CWnd
    - CWndContainer
      - CScroll
        - CScrollH

Résultat du code fournit ci-dessous :

Méthodes de Classe

© 2000-2024, MetaQuotes Ltd.

Contrôles dépendants	
CreateInc	Cré le bouton d'incrémentation de la barre de défilement
CreateDec	Cré le bouton de décrémentation de la barre de défilement
CreateThumb	Crée le bouton de l'ascenseur (pouvant être déplacé)
Gestionnaires d'évènements internes	
OnResize	Gestionnaire d'évènement "Resize"
OnChangePos	Gestionnaire d'évènement "ChangePosition"
Gestionnaires d'évènement de Drag n' Drop	
OnThumbDragStart	Gestionnaire d'évènement "ThumbDragStart"
OnThumbDragProcess	Gestionnaire d'évènement "ThumbDragProcess"
OnThumbDragEnd	Gestionnaire d'évènement "ThumbDragEnd"
Position	
CalcPos	Retourne la position de la barre de défilement par ses coordonnées

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Méthodes héritées de la classe CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Exemple de création d'un panneau avec une barre de défilement horizontale :

```
//+------------------------------------------------------------------+
//|                                              ControlsScrollH.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
```

```
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CSc
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollH          m_scroll_v;                     // Objet CScrollH

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crée des contrôles dépendants
   bool              CreateScrollsH(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateScrollsH())
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée l'objet CScrollsH                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollsH(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+3*BUTTON_WIDTH;
   int y2=y1+18;
//--- création
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollsH",m_subwin,x1,y1,x2,y2))
      return(false);
//--- définit la barre de défilement
   m_scroll_v.MinPos(0);
//--- définit la barre de défilement
   m_scroll_v.MaxPos(10);
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!Add(m_scroll_v))
      return(false);
   Comment("Position de la barre de défilement ",m_scroll_v.CurrPos());
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position de la barre de défilement ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position de la barre de défilement ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert       |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                       |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprime le commentaire
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // identifiant de l'évènement
                  const long& lparam,    // paramètre d'évènement de type long
                  const double& dparam,  // paramètre d'évènement de type double
                  const string& sparam)  // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

CreateInc

Crée le bouton d'incrémentation du contrôle.

```
virtual bool  CreateInc()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateDec

Crée le bouton de décrémentation du contrôle.

```
virtual bool  CreateDec()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateThumb

Crée l'ascenseur (pouvant être déplacé) du contrôle.

```
virtual bool  CreateThumb()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChangePos

Le gestionnaire d'évènement "ChangePos" (changement de position) du contrôle.

```
virtual bool  OnChangePos()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragStart

Le gestionnaire d'évènement "ThumbDragStart" (début du déplacement de l'ascenceur) du contrôle.

```
virtual bool  OnThumbDragStart()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragStart" est généré au démarrage du déplacement de l'ascenseur.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragProcess

Le gestionnaire d'évènement "ThumbDragProcess" du contrôle.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // Coordonnée X
   const int  y       // Coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X actuelle du curseur de la souris.

y

[in] Coordonnée Y actuelle du curseur de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragProcess" est généré lorsque le contrôle de la barre de défilement (l'ascenseur) est déplacé.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragEnd

Le gestionnaire d'évènement "ThumbDragEnd" (fin du déplacement de l'ascenseur) du contrôle.

```
virtual bool  OnThumbDragEnd()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragEnd" est généré lorsque le déplacement du contrôle de la barre de défilement (l'ascenseur) est terminé.

© 2000-2024, MetaQuotes Ltd.

CalcPos

Retourne la position de la barre de défilement par ses coordonnées

```
virtual int  CalcPos(
   const int      coord      // coordonnée
   )
```

Paramètres

coord

[in] Coordonnée de la barre de défilement.

Valeur de retour

Position de la barre de défilement.

© 2000-2024, MetaQuotes Ltd.

CWndClient

CWndClient est une classe du contrôle complexe de "Zone client" (avec des contrôles inter-dépendants). C'est la classe de base pour la création des zones de défilement des barres de défilement.

Description

CWndClient implémente la fonctionnalité de création des zone client avec des barres de défilement.

Déclaration

```
class CWndClient : public CWndContainer
```

Titre

```
#include <Controls\WndClient.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndContainer
      CWndClient

Descendants directs

CCheckGroup, CListView, CRadioGroup

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaire d'évènement graphique.	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
ColorBackground	Définit la couleur d'arrière plan.
ColorBorder	Définit la couleur de la bordure.
BorderType	Définit le type de bordure.
Paramètres	
VScrolled	Retourne/Définit le flag indiquant que la barre de défilement verticale est utilisée

© 2000-2024, MetaQuotes Ltd.

Création	
HScrolled	Retourne/Définit le flag indiquant que la barre de défilement horizontale est utilisée
Contrôles dépendants	
CreateBack	Cée l'arrière plan de la barre de défilement
CreateScrollV	Crée la barre de défilement verticale
CreateScrollH	Crée la barre de défilement horizontale
Gestionnaires d'évènements internes	
OnResize	Gestionnaire d'évènement "Resize"
Gestionnaires d'évènements des contrôles dépendants	
OnVScrollShow	Gestionnaire (virtuel) de l'évènement "Show" du contrôle dépendant VScroll
OnVScrollHide	Gestionnaire (virtuel) de l'évènement "Hide" du contrôle dépendant VScroll
OnHScrollShow	Gestionnaire (virtuel) de l'évènement "Show" du contrôle dépendant HScroll
OnHScrollHide	Gestionnaire (virtuel) de l'évènement "Hide" du contrôle dépendant HScroll
OnScrollLineDown	Gestionnaire (virtuel) de l'évènement "ScrollLineDown" du contrôle dépendant VScroll
OnScrollLineUp	Gestionnaire (virtuel) de l'évènement "ScrollLineUp" du contrôle dépendant VScroll
OnScrollLineLeft	Gestionnaire (virtuel) de l'évènement "ScrollLineLeft" du contrôle dépendant HScroll
OnScrollLineRight	Gestionnaire (virtuel) de l'évènement "ScrollLineRight" du contrôle dépendant HScroll
Resize	
Rebound	Définit les nouvelles coordonnées du contrôle en utilisant les coordonnées de la classe CRect

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CWndClient.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

ColorBackground

Définit la couleur d'arrière plan du contrôle.

```
bool  ColorBackground(
   const color  value      // nouvelle couleur
   )
```

Paramètres

value

[in] Nouvelle couleur d'arrière plan du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ColorBorder

Définit la couleur de bordure du contrôle.

```
bool  ColorBorder(
   const color  value      // couleur
   )
```

Paramètres

value

[in] Nouvelle couleur de bordure du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

BorderType

Définit le type de bordure du contrôle.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // type de bordure
   )
```

Paramètres

type

[in] Type de bordure du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

VScrolled (Méthode "Get")

Retourne/Définit le flag indiquant que la barre de défilement verticale est utilisée.

```
bool  VScrolled()
```

Valeur de retour

vrai si la barre de défilement verticale est utilisée, faux sinon.

VScrolled (Méthode "Set")

Définit le flag indiquant que la barre de défilement verticale est utilisée.

```
bool  VScrolled(
   const bool  flag      // flag
   )
```

Paramètres

flag

[in] Flag.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

HScrolled (Méthode "Get")

Retourne/Définit le flag indiquant que la barre de défilement horizontale est utilisée.

```
bool  HScrolled()
```

Valeur de retour

vrai si la barre de défilement horizontal est utilisée, faux sinon.

HScrolled (Méthode "Set")

Définit le flag indiquant que la barre de défilement horizontale est utilisée.

```
bool  HScrolled(
   const bool  flag      // flag
   )
```

Paramètres

flag

[in] Flag.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateBack

Crée le bouton d'arrière plan du contrôle.

```
virtual bool  CreateBack()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateScrollV

Crée la barre de défilement verticale

```
virtual bool  CreateScrollV()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateScrollH

Crée la barre de défilement horizontale.

```
virtual bool  CreateScrollH()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnVScrollShow

Le gestionnaire d'évènement "VScrollShow" (affichage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnVScrollHide

Le gestionnaire d'évènement "VScrollHide" (masquage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnHScrollShow

Le gestionnaire d'évènement "HScrollShow" (affichage de la barre de défilement horizontale) du contrôle.

```
virtual bool  OnHScrollShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnHScrollHide

Le gestionnaire d'évènement "HScrollHide" (masquage de la barre de défilement horizontale) du contrôle.

```
virtual bool  OnHScrollHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineDown

Le gestionnaire d'évènement "ScrollLineDown" (défilement vertical vers le bas) du contrôle.

```
virtual bool  OnScrollLineDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineUp

Le gestionnaire d'évènement "ScrollLineUp" (défilement vertical vers le haut) du contrôle.

```
virtual bool  OnScrollLineUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineLeft

Le gestionnaire d'évènement "ScrollLineLeft" (défilement vers la gauche) du contrôle.

```
virtual bool  OnScrollLineLeft()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineRight

Le gestionnaire d'évènement "ScrollLineRight" (défilement vers la droite) du contrôle.

```
virtual bool  OnScrollLineRight()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2024, MetaQuotes Ltd.

ReBound

Définit les nouvelles coordonnées du contrôle utilisant les coordonnées de la classe CRect.

```
void  ReBound(
   const & CRect  rect      // Classe CRect
   )
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CListView

La classe CListView est une classe de contrôle complexe de ListView (avec des contrôles inter-dépendants).

Description

La classe CListView encapsule les fonctionnalités de contrôle de liste.

Déclaration

```
class CListView : public CWndClient
```

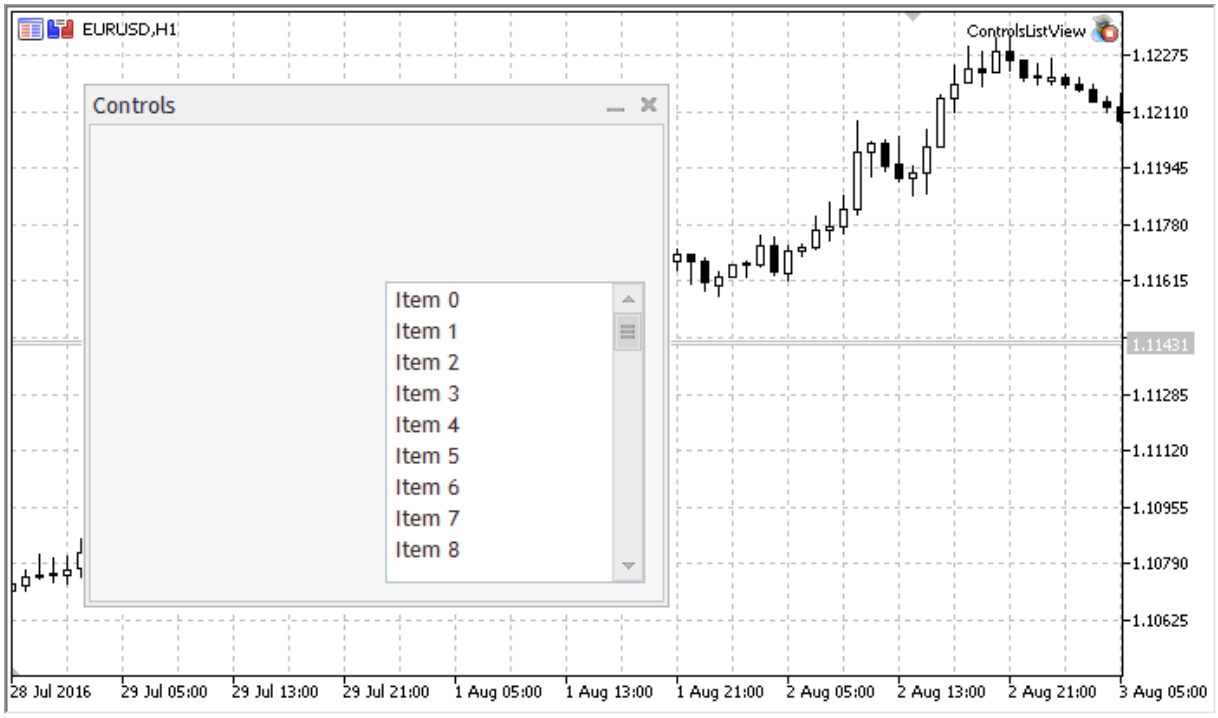
Titre

```
#include <Controls\ListView.mqh>
```

Hiérarchie d'héritage

COBJECT
CWnd
CWndContainer
CWndClient
CListView

Résultat du code fournit ci-dessous :



Méthodes de Classe

© 2000-2024, MetaQuotes Ltd.

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Paramètres	
TotalView	Définit le nombre d'éléments affichés dans le contrôle
Ajout/Suppression	
AddItem	Ajoute un élément
Données	
Select	Sélectionne l'élément courant par son index
SelectByText	Séléection l'élément courant par son texte
SelectByValue	Séléection l'élément courant par sa valeur
Données en lecture seule	
Value	Retourne la valeur de l'élément courant de la liste
Contrôles dépendants	
CreateRow	Crée une ligne dans la liste ListView
Gestionnaires d'évènements internes	
OnResize	Gestionnaire (virtuel) de l'évènement "Resize"
Gestionnaires d'évènements des contrôles dépendants	
OnVScrollShow	Gestionnaire (virtuel) de l'évènement "Show" du contrôle dépendant VScroll
OnVScrollHide	Gestionnaire (virtuel) de l'évènement "Hide" du contrôle dépendant VScroll
OnScrollLineDown	Gestionnaire (virtuel) de l'évènement "ScrollLineDown" du contrôle dépendant VScroll
OnScrollLineUp	Gestionnaire (virtuel) de l'évènement "ScrollLineUp" du contrôle dépendant VScroll
OnItemClick	Gestionnaire (virtuel) de l'évènement "ItemClick"
Rafraîchissement	
Redraw	Redessine le contrôle

© 2000-2024, MetaQuotes Ltd.

Création	
RowState	Définit l'état de la ligne spécifiée
CheckView	Vérifie la visibilité de la ligne spécifiée

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

Méthodes héritées de la classe CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Exemple de création d'un panneau avec un contrôle de type list view :

```
//+------------------------------------------------------------------+
//|                                              ControlsListView.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CLis
#include <Controls\Dialog.mqh>
#include <Controls\ListView.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
```

```
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles    |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CListView         m_list_view;                     // Objet CListView

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crée des contrôles dépendants
   bool              CreateListView(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnChangeListView(void);
  };
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_list_view,OnChangeListView)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateListView())
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée l'élément "ListView"                                        |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateListView(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT+GROUP_WIDTH+2*CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+2*CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+LIST_HEIGHT-CONTROLS_GAP_Y;
//--- création
   if(!m_list_view.Create(m_chart_id,m_name+"ListView",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_list_view))
      return(false);
//--- remplissage
   for(int i=0;i<16;i++)
      if(!m_list_view.AddItem("Item "+IntegerToString(i)))
         return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeListView(void)
  {
   Comment(__FUNCTION__+" \""+m_list_view.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert                            |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprime le commentaire
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre d'évènement de type long
                  const double& dparam, // paramètre d'évènement de type double
                  const string& sparam) // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CListView.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

TotalView

Définit le nombre d'éléments affichés dans le contrôle

```
bool  TotalView(
   const int  value      // éléments affichés
   )
```

Paramètres

value

[in] Le nombre d'éléments affichés dans le contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Le nombre d'éléments affichés ne peut être spécifié qu'une seule fois.

© 2000-2024, MetaQuotes Ltd.

AddItem

Ajoute un élément au contrôle.

```
bool  AddItem(
   const string  item,      // texte
   const long    value      // valeur
   )
```

Paramètres

item

[in] Texte.

value

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Select

Sélectionne l'élément courant par son index.

```
bool  Select(
   const int  index     // index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

SelectByText

Séléection l'élément courant par son texte

```
bool  SelectByText(
   const string  text     // texte
   )
```

Paramètres

text

[in] Texte.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

SelectByValue

Séléection l'élément courant par sa valeur

```
bool  SelectByValue(
   const long  value     // valeur
   )
```

Paramètres

value

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Value

Retourne la valeur de l'élément courant de la liste.

```
long  Value()
```

Valeur de retour

La valeur de l'élément courant de la liste.

© 2000-2024, MetaQuotes Ltd.

CreateRow

Crée une ligne dans le contrôle ListView

```
bool  CreateRow(
   const int  index     // index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnVScrollShow

Le gestionnaire d'évènement "VScrollShow" (affichage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnVScrollHide

Le gestionnaire d'évènement "VScrollHide" (masquage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineDown

Le gestionnaire d'évènement "ScrollLineDown" (défilement vertical vers le bas) du contrôle.

```
virtual bool  OnScrollLineDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineUp

Le gestionnaire d'évènement "ScrollLineUp" (défilement vertical vers le haut) du contrôle.

```
virtual bool  OnScrollLineUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnItemClick

Le gestionnaire d'évènement "ItemClick" (clic de souris sur une ligne) du contrôle.

```
virtual bool  OnItemClick()
   const int    index     // index de la ligne
   )
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Redraw

Redessine le contrôle.

```
bool  Redraw()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

RowState

Définit l'état de la ligne spécifiée.

```
bool  RowState(
   const int   index      // index
   const bool  select     // état
   )
```

Paramètres

index

[in] Index de la ligne.

select

[in] Etat de la ligne.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CheckView

Vérifie la visibilité de la ligne spécifiée.

```
bool  CheckView()
```

Valeur de retour

vrai - si la ligne sélectionnée est visible, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CComboBox

La classe CComboBox est une classe de contrôle complexe de ComboBox (avec des contrôles inter-dépendants).

Description

Le contrôle ComboBox est constitué d'une liste et d'un contrôle statique et est prévu pour la sélection. La partie 'liste' du contrôle se déroule lorsque l'utilisateur clique sur la flèche située à côté du contrôle

Déclaration

```
class CComboBox : public CWndContainer
```

Titre

```
#include <Controls\ComboBox.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndContainer
      CComboBox

Résultat du code fournit ci-dessous :

Méthodes de Classe

© 2000-2024, MetaQuotes Ltd.

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Add	
AddItem	Ajoute un élément
Paramètres	
ListViewItems	Définit le nombre d'éléments affichés dans le contrôle
Données	
Select	Sélectionne l'élément courant par son index
SelectByText	Séléection l'élément courant par son texte
SelectByValue	Séléection l'élément courant par sa valeur
Données en lecture seule	
Value	Retourne la valeur de l'élément courant de la liste
Contrôles dépendants	
CreateEdit	Crée un contrôle dépendant (zone d'édition)
CreateButton	Crée un contrôle dépendant (bouton)
CreateList	Crée un contrôle dépendant (liste)
Gestionnaires d'évènements des contrôles dépendants	
OnClickEdit	Gestionnaire (virtuel) de l'évènement "ClickEdit"
OnClickButton	Gestionnaire (virtuel) de l'évènement "ClickButton"
OnChangeList	Gestionnaire (virtuel) de l'évènement "ChangeList"
Affichage/Masquage	
ListShow	Affiche la liste des éléments
ListHide	Cache la liste des éléments

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Hide

Example of creating a panel with Combobox control:

```
//+------------------------------------------------------------------+
//|                                              ControlsComboBox.mq5 |
//|                        Copyright 2015, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2015, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CCon
#include <Controls\Dialog.mqh>
#include <Controls\ComboBox.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles    |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
```

© 2000-2024, MetaQuotes Ltd.

```
  {
private:
   CComboBox         m_combo_box;;                    // Objet CComboBox

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crée des contrôles dépendants
   bool              CreateComboBox(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnChangeComboBox(void);
  };
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_combo_box,OnChangeComboBox)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateComboBox())
      return(false);
//--- succès
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Crée l'élément "ComboBox"                                        |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateComboBox(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- création
   if(!m_combo_box.Create(m_chart_id,m_name+"ComboBox",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_combo_box))
      return(false);
//--- remplissage
   for(int i=0;i<16;i++)
      if(!m_combo_box.ItemAdd("Item "+IntegerToString(i)))
         return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeComboBox(void)
  {
   Comment(__FUNCTION__+" \""+m_combo_box.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert        |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre d'évènement de type long
                  const double& dparam, // paramètre d'évènement de type double
                  const string& sparam) // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CComboBox.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

AddItem

Ajoute un élément au contrôle.

```
bool  AddItem(
   const string  item,      // texte
   const long    value      // valeur
   )
```

Paramètres

item

[in] Texte.

value=0

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ListViewItems

Définit le nombre d'éléments affichés dans le contrôle CComboBox.

```
void  ListViewItems(
   const int    value     // nombre d'éléments
   )
```

Paramètres

value

[in] Nombre d'éléments dans la liste.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Select

Sélectionne l'élément courant par son index.

```
bool  Select(
   const int  index     // index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

SelectByText

Séléection l'élément courant par son texte

```
bool  SelectByText(
   const string  text     // texte
   )
```

Paramètres

text

[in] Texte de l'élément.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

SelectByValue

Séléection l'élément courant par sa valeur

```
bool  SelectByValue(
   const long  value     // valeur
   )
```

Paramètres

value

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Value

Retourne la valeur de l'élément courant de la liste.

```
long  Value()
```

Valeur de retour

La valeur de l'élément courant de la liste.

© 2000-2024, MetaQuotes Ltd.

CreateEdit

Crée un contrôle dépendant (zone d'édition)

```
virtual bool  CreateEdit()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateButton

Crée un contrôle dépendant (bouton).

```
virtual bool  CreateButton()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateList

Crée un contrôle dépendant (liste)

```
virtual bool  CreateList()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnClickEdit

Le gestionnaire d'évènement "ClickEdit" (clic de souris sur la zone d'édition) du contrôle.

```
virtual bool  OnClickEdit()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnClickButton

Le gestionnaire d'évènement "ClickButton" (clic de souris sur le bouton) du contrôle.

```
virtual bool  OnClickButton()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChangeList

Le gestionnaire d'évènement "ChangeList" (changement dans la liste).

```
virtual bool  OnChangeList()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

ListShow

Affiche la liste des éléments.

```
virtual bool  ListShow()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ListHide

Cache la liste des éléments.

```
virtual bool  ListHide()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CCheckBox

La classe CCheckBox est une classe de contrôle complexe de CheckBox.

Description

Le contrôle CCheckBox affiche une case à cocher permettant à l'utilisateur de sélectionner une condition vraie ou fausse.

Déclaration

```
class CCheckBox : public CWndContainer
```

Titre

```
#include <Controls\CheckBox.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndContainer
      CCheckBox

Résultat du code fournit ci-dessous :

Méthodes de Classe

Création	
Create	Crée le contrôle

© 2000-2024, MetaQuotes Ltd.

Création	
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
Texte	Retourne/Définit le texte de l'étiquette associée au contrôle
Color	Retourne/Définit la couleur du texte de l'étiquette associée au contrôle
Etat	
Checked	Retourne/Définit une valeur indiquant si le contrôle est coché ou pas
Données	
Value	Retourne/Définit la valeur associée au contrôle
Contrôles dépendants	
CreateButton	Crée un contrôle dépendant (bouton)
CreateLabel	Crée un contrôle dépendant (étiquette)
Gestionnaires d'évènements des contrôles dépendants	
ClickButton	Gestionnaire (virtuel) de l'évènement "ClickButton"
ClickLabel	Gestionnaire (virtuel) de l'évènement "ClickLabel"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Example of creating a panel with Checkbox control:

```
//+------------------------------------------------------------------+
//|                                              ControlsCheckBox.mq5 |
```

© 2000-2024, MetaQuotes Ltd.

```
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CChe
#include <Controls\Dialog.mqh>
#include <Controls\CheckBox.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CCheckBox         m_check_box1;                    // Objet CCheckBox
   CCheckBox         m_check_box2;                    // Objet CCheckBox
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
protected:
   //--- crée des contrôles dépendants
   bool              CreateCheckBox1(void);
   bool              CreateCheckBox2(void);
```

```
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnChangeCheckBox1(void);
   void              OnChangeCheckBox2(void);
  };
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_box1,OnChangeCheckBox1)
ON_EVENT(ON_CHANGE,m_check_box2,OnChangeCheckBox2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateCheckBox1())
      return(false);
   if(!CreateCheckBox2())
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée l'élément "CheckBox"                                        |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox1(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
```

© 2000-2024, MetaQuotes Ltd.

```
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- création
   if(!m_check_box1.Create(m_chart_id,m_name+"CheckBox1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box1.Text("CheckBox1"))
      return(false);
   if(!m_check_box1.Color(clrBlue))
      return(false);
   if(!Add(m_check_box1))
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée l'élément "CheckBox"                                        |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox2(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT+GROUP_WIDTH+CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- création
   if(!m_check_box2.Create(m_chart_id,m_name+"CheckBox2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box2.Text("CheckBox2"))
      return(false);
   if(!m_check_box2.Color(clrBlue))
      return(false);
   if(!Add(m_check_box2))
      return(false);
   m_check_box2.Checked(true);
   Comment(__FUNCTION__+" : Coché="+IntegerToString(m_check_box2.Checked()));
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox1(void)
  {
   Comment(__FUNCTION__+" : Coché="+IntegerToString(m_check_box1.Checked()));
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox2(void)
  {
   Comment(__FUNCTION__+" : Coché="+IntegerToString(m_check_box2.Checked()));
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert       |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre d'évènement de type long
                  const double& dparam, // paramètre d'évènement de type double
                  const string& sparam) // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CCheckBox.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

Text (Méthode "Get")

Retourne le texte de l'étiquette associée au contrôle.

```
string  Text()
```

Valeur de retour

Texte de l'étiquette.

Text (Méthode "Set")

Définit le texte de l'étiquette associée au contrôle.

```
bool  Text(
   const string  value      // texte
   )
```

Paramètres

value

[in] Nouveau texte de l'étiquette.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Color (Méthode "Get")

Retourne la couleur de l'étiquette associée au contrôle.

```
color Color() const
```

Valeur de retour

Couleur de l'étiquette.

Color (Méthode "Set")

Définit la couleur de l'étiquette associée au contrôle.

```
bool Color(
   const color value      // couleur
   )
```

Paramètres

value

[in] Nouvelle couleur d'étiquette.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Checked (Méthode "Get")

Retourne l'état du contrôle

```
bool  Checked()  const
```

Valeur de retour

Etat du contrôle.

Checked (Méthode "Set")

Définit l'état du contrôle.

```
bool  Checked(
   const bool  flag      // état
   )
```

Paramètres

flag

[in] Nouvel état.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Value (Méthode "Get")

Retourne la valeur associée au contrôle.

```
int  Value()  const
```

Valeur de retour

La valeur associée au contrôle.

Value (Méthode "Set")

Définit la valeur associée au contrôle.

```
void  Value(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CreateButton

Crée un contrôle dépendant (bouton).

```
virtual bool  CreateButton()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateLabel

Crée un contrôle dépendant (étiquette)

```
virtual bool  CreateLabel()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnClickButton

Le gestionnaire d'évènement "ClickButton" (clic de souris sur le bouton) du contrôle.

```
virtual bool  OnClickButton()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnClickLabel

Le gestionnaire d'évènement "ClickLabel" (clic de souris sur l'étiquette) du contrôle.

```
virtual bool  OnClickLabel()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CCheckGroup

La classe CCheckGroup est une classe de contrôle complexe de CheckGroup (avec des contrôles inter-dépendants).

Description

La classe CCheckGroup permet de créer des contrôles permettant d'afficher et d'éditer des flags.

Déclaration

```
class CCheckGroup : public CWndClient
```

Titre

```
#include <Controls\CheckGroup.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndContainer
      CWndClient
        CCheckGroup

Résultat du code fournit ci-dessous :

Méthodes de Classe

© 2000-2024, MetaQuotes Ltd.

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement de tous les évènements graphiques
Add	
AddItem	Ajoute un nouvel élément
Données en lecture seule	
Value	Retourne la valeur associée au contrôle
Contrôles dépendants	
CreateButton	Crée un nouvel élément CCheckBox
Gestionnaires d'évènements des contrôles dépendants	
OnVScrollShow	Gestionnaire (virtuel) de l'évènement "Show" du contrôle dépendant VScroll
OnVScrollHide	Gestionnaire (virtuel) de l'évènement "Hide" du contrôle dépendant VScroll
OnScrollLineDown	Gestionnaire (virtuel) de l'évènement "ScrollLineUp" du contrôle dépendant VScroll
OnScrollLineUp	Gestionnaire (virtuel) de l'évènement "ScrollLineDown" du contrôle dépendant VScroll
OnChangeItem	Gestionnaire (virtuel) de l'évènement "ChangeItem" du contrôle dépendant VScroll
Rafraichissement	
Redraw	Redessine le groupe
RowState	Définit l'état de l'élément spécifié

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Méthodes héritées de la classe CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Example of creating a panel with Checkbox group control:

```
//+------------------------------------------------------------------+
//|                                          ControlsCheckGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CChe
#include <Controls\Dialog.mqh>
#include <Controls\CheckGroup.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
```

```
   CCheckGroup       m_check_group;                    // Objet CCheckGroup

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crée des contrôles dépendants
   bool              CreateCheckGroup(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnChangeCheckGroup(void);
  };
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_group,OnChangeCheckGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateCheckGroup())
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée l'élément "CheckGroup"                                      |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckGroup(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+CHECK_HEIGHT;
//--- création
   if(!m_check_group.Create(m_chart_id,m_name+"CheckGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_check_group))
      return(false);
//--- remplissage
   for(int i=0;i<5;i++)
      if(!m_check_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_check_group.Check(0,1<<0);
   m_check_group.Check(2,1<<2);
   Comment(__FUNCTION__+" : Valeur="+IntegerToString(m_check_group.Value()));
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckGroup(void)
  {
   Comment(__FUNCTION__+" : Valeur="+IntegerToString(m_check_group.Value()));
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert       |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(ChartID(),"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
```

© 2000-2024, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre d'évènement de type long
                  const double& dparam, // paramètre d'évènement de type double
                  const string& sparam) // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CCheckGroup.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

AddItem

Ajoute un élément au contrôle.

```
bool  AddItem(
   const string  item,      // texte
   const long    value      // valeur
   )
```

Paramètres

item

[in] Texte.

value=0

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Value

Retourne la valeur associée au contrôle.

```
long  Value()
```

Valeur de retour

La valeur associée au contrôle.

Note

La valeur dépend des états de tous les éléments du groupe CCheckGroup.

© 2000-2024, MetaQuotes Ltd.

CreateButton

Crée une nouvelle instance de la classe CCheckBox à l'index spécifié.

```
bool  CreateButton(
   int  index      // index
   )
```

Paramètres

index

[in] Index du nouvel élément au sein du CCheckGroup.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnVScrollShow

Le gestionnaire d'évènement "VScrollShow" (affichage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnVScrollHide

Le gestionnaire d'évènement "VScrollHide" (masquage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineDown

Le gestionnaire d'évènement "ScrollLineDown" (défilement vertical vers le bas) du contrôle.

```
virtual bool  OnScrollLineDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineUp

Le gestionnaire d'évènement "ScrollLineUp" (défilement vertical vers le haut) du contrôle.

```
virtual bool  OnScrollLineUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChangeItem

Le gestionnaire d'évènement "ChangeItem" du contrôle.

```
virtual bool  OnChangeItem(
   const int  index      // index
   )
```

Paramètres

index

[in] Index de l'élément modifié.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Redraw

Redessine le contrôle.

```
bool  Redraw()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

RowState

Définit l'état de l'élément spécifié.

```
bool  RowState(
   const int   index,       // index de l'élément
   const bool  select       // état
   )
```

Paramètres

index

[in] Index de l'élément à modifier.

select

[in] Nouvel état.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CRadioButton

La classe CRadioButton est une classe de contrôle complexe de RadioButton.

Description

La classe CRadioButton elle-même n'est pas utilisée, elle est utilisée pour la création des éléments de type CRadioGroup.

Déclaration

```
class CRadioButton : public CWndContainer
```

Titre

```
#include <Controls\RadioButton.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndContainer
      CRadioButton

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement de tous les évènements graphiques
Propriétés	
Texte	Retourne/Définit le texte de l'étiquette associée au contrôle
Color	Retourne/Définit la couleur du texte de l'étiquette associée au contrôle
Etat	
Etat	Retourne/Définit l'état
Contrôles dépendants	
CreateButton	Crée un bouton
CreateLabel	Crée une étiquette
Gestionnaires d'évènements des contrôles dépendants	

© 2000-2024, MetaQuotes Ltd.

Création	
OnClickButton	Gestionnaire (virtuel) de l'évènement "ClickButton"
OnClickLabel	Gestionnaire (virtuel) de l'évènement "ClickLabel"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000-2024, MetaQuotes Ltd.

Create

Crée un contrôle CRadioButton.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

Text (Méthode "Get")

Retourne le texte de l'étiquette associée au contrôle.

```
string  Text()
```

Valeur de retour

Texte de l'étiquette.

Text (Méthode "Set")

Définit le texte de l'étiquette associée au contrôle.

```
bool  Text(
   const string  value      // texte
   )
```

Paramètres

value

[in] Nouveau texte de l'étiquette.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Color (Méthode "Get")

Retourne la couleur de l'étiquette associée au contrôle.

```
color Color() const
```

Valeur de retour

Couleur de l'étiquette.

Color (Méthode "Set")

Définit la couleur de l'étiquette associée au contrôle.

```
bool Color(
   const color value      // couleur
   )
```

Paramètres

value

[in] Nouvelle couleur d'étiquette.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

State (Méthode "Get")

Retourne l'état du bouton.

```
bool State() const
```

Valeur de retour

Etat du bouton.

State (Méthode "Set")

Définit l'état du bouton.

```
bool State(
   const bool flag      // flag
   )
```

Paramètres

flag

[in] Nouvel état du bouton.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateButton

Crée un bouton.

```
virtual bool  CreateButton()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateLabel

Crée une étiquette.

```
virtual bool  CreateLabel()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnClickButton

Le gestionnaire d'évènement "ClickButton" (clic de souris) du contrôle.

```
virtual bool  OnClickButton()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnClickLabel

Le gestionnaire d'évènement "ClickLabel" (clic de souris sur l'étiquette) du contrôle.

```
virtual bool  OnClickLabel()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CRadioGroup

La classe CRadioGroup est une classe de contrôle complexe de RadioGroup (avec des contrôles inter-dépendants).

Description

La classe CRadioGroup permet à l'utilisateur de sélectionner une seule option d'un groupe de choix parmi différents contrôles CRadioButton.

Déclaration

```
class CRadioGroup : public CWndClient
```

Titre

```
#include <Controls\RadioGroup.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndContainer
      CWndClient
        CRadioGroup

Résultat du code fournit ci-dessous :

Méthodes de Classe

© 2000-2024, MetaQuotes Ltd.

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Add	
AddItem	Ajoute un nouvel élément
Données en lecture seule	
Value	Retourne la valeur associée au contrôle
Contrôles dépendants	
CreateButton	Crée un nouvel élément CRadioButton
Gestionnaires d'évènements des contrôles dépendants	
OnVScrollShow	Gestionnaire (virtuel) de l'évènement "Show" du contrôle dépendant VScroll
OnVScrollHide	Gestionnaire (virtuel) de l'évènement "Hide" du contrôle dépendant VScroll
OnScrollLineDown	Handler (virtuel) de l'évènement "ScrollLineDown" du contrôle dépendant VScroll
OnScrollLineUp	Gestionnaire (virtuel) de l'évènement "ScrollLineUp" du contrôle dépendant VScroll
OnChangeItem	Gestionnaire (virtuel) de l'évènement "ChangeItem"
Rafraîchissement	
Redraw	Redessine le groupe d'éléments
RowState	Définit l'état de l'élément spécifié
Select	Sélectionne l'élément courant

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Méthodes héritées de la classe CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Exemple de création d'un panneau avec un groupe de boutons radios :

```
//+------------------------------------------------------------------+
//|                                          ControlsRadioGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CRad
#include <Controls\Dialog.mqh>
#include <Controls\RadioGroup.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
```

```
  {
private:
   CRadioGroup       m_radio_group;                    // Objet CRadioGroup

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crée des contrôles dépendants
   bool              CreateRadioGroup(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnChangeRadioGroup(void);
  };
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_radio_group,OnChangeRadioGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateRadioGroup())
      return(false);
//--- succès
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Crée l'élément "RadioGroup"                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateRadioGroup(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+RADIO_HEIGHT;
//--- création
   if(!m_radio_group.Create(m_chart_id,m_name+"RadioGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_radio_group))
      return(false);
//--- remplissage
   for(int i=0;i<3;i++)
      if(!m_radio_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_radio_group.Value(1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeRadioGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert       |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
```

© 2000-2024, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprime le commentaire
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identifiant de l'évènement
                  const long& lparam,   // paramètre d'évènement de type long
                  const double& dparam, // paramètre d'évènement de type double
                  const string& sparam) // paramètre d'évènement de type string
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CRadioGroup.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

AddItem

Ajoute un élément au contrôle.

```
bool  AddItem(
   const string  item,      // texte
   const long    value=0    // valeur
   )
```

Paramètres

item

[in] Texte.

value=0

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Value

Retourne la valeur associée au contrôle.

```
long  Value()
```

Valeur de retour

La valeur associée au contrôle.

Note

La valeur dépend de l'état de tous les éléments CRadioButton du contrôle CRadioGroup.

© 2000-2024, MetaQuotes Ltd.

CreateButton

Crée une nouvelle instance de classe CRadioButton à l'index spécifié.

```
bool  CreateButton(
   const int  index      // index
   )
```

Paramètres

index

[in] Index du nouvel élément dans le groupe CRadioGroup.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnVScrollShow

Le gestionnaire d'évènement "VScrollShow" (affichage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnVScrollHide

Le gestionnaire d'évènement "VScrollHide" (masquage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineDown

Le gestionnaire d'évènement "ScrollLineDown" (défilement vertical vers le bas) du contrôle.

```
virtual bool  OnScrollLineDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineUp

Le gestionnaire d'évènement "ScrollLineUp" (défilement vertical vers le haut) du contrôle.

```
virtual bool  OnScrollLineUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChangeItem

Le gestionnaire d'évènement "ChangeItem" du contrôle.

```
virtual bool  OnChangeItem(
   const int  index      // index
   )
```

Paramètres

index

[in] Index de l'élément modifié.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

Redraw

Redessine le contrôle.

```
bool  Redraw()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

RowState

Définit l'état de l'élément spécifié.

```
bool  RowState(
   const int   index,       // index de l'élément
   const bool  select       // état
   )
```

Paramètres

index

[in] Index de l'élément à modifier.

select

[in] Nouvel état.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Select

Sélectionne l'élément courant.

```
void  Select(
   const int  index      // index
   )
```

Paramètres

index

[in] Index de l'élément à sélectionner.

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CSpinEdit

La classe CSpinEdit est la classe du contrôle complexe SpinEdit (avec des contrôles inter-dépendants).

Description

La classe CSpinEdit permet la création de contrôles permettant d'éditer des valeurs de type integer. La valeur est incrémentée automatiquement lors de l'appui sur le bouton haut, et décrémentée en appuyant sur le bouton bas.

Déclaration

```
class CSpinEdit : public CWndContainer
```

Titre

```
#include <Controls\SpinEdit.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndContainer
      CSpinEdit

Résultat du code fournit ci-dessous :

Méthodes de Classe

© 2000-2024, MetaQuotes Ltd.

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
MinValue	Retourne/Définit la position minimale permise
MaxValue	Retourne/Définit la position maximale permise
Etat	
Value	Retourne/Définit la valeur courante
Contrôles dépendants	
CreateEdit	Crée un contrôle dépendant (zone d'édition)
CreateInc	Crée un contrôle dépendant (bouton haut)
CreateDec	Crée un contrôle dépendant (bouton bas)
Gestionnaires d'évènements des contrôles dépendants	
OnClickInc	Gestionnaire (virtuel) de l'évènement "ClickInc"
OnClickDec	Gestionnaire (virtuel) de l'évènement "ClickDec"
Gestionnaires d'évènements internes	
OnChangeValue	Gestionnaire (virtuel) de l'évènement "ChangeValue"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Exemple de création d'un panneau avec un contrôle spin edit :

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              ControlsSpinEdit.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Panneaux de Contrôle et Dialogues. Classe de démonstration CSpi
#include <Controls\Dialog.mqh>
#include <Controls\SpinEdit.mqh>
//+------------------------------------------------------------------+
//| définition                                                       |
//+------------------------------------------------------------------+
//--- placement
#define INDENT_LEFT                         (11)      // indentation depuis la gauche
#define INDENT_TOP                          (11)      // indentation depuis la sommet
#define INDENT_RIGHT                        (11)      // indentation depuis la droite
#define INDENT_BOTTOM                       (11)      // indentation depuis le fond (a
#define CONTROLS_GAP_X                      (5)       // coordonnée X du gap
#define CONTROLS_GAP_Y                      (5)       // coordonnée Y du gap
//--- pour les boutons
#define BUTTON_WIDTH                        (100)     // largeur
#define BUTTON_HEIGHT                       (20)      // hauteur
//--- pour la zone d'indication
#define EDIT_HEIGHT                         (20)      // hauteur
//--- pour les contrôles de type groupe
#define GROUP_WIDTH                         (150)     // largeur
#define LIST_HEIGHT                         (179)     // hauteur
#define RADIO_HEIGHT                        (56)      // hauteur
#define CHECK_HEIGHT                        (93)      // hauteur
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage : dialogue principale de l'application des Contrôles   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CSpinEdit         m_spin_edit;                      // Objet CSpinEdit

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- création
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- gestionnaire d'évènement du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
```

```
   //--- crée des contrôles dépendants
   bool              CreateSpinEdit(void);
   //--- gestionnaires des évènements de contrôle dépendants
   void              OnChangeSpinEdit(void);
  };
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
   ON_EVENT(ON_CHANGE,m_spin_edit,OnChangeSpinEdit)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructeur                                                     |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructeur                                                      |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Création                                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- crée des contrôles dépendants
   if(!CreateSpinEdit())
      return(false);
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Crée l'élément "SpinEdit"                                        |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateSpinEdit(void)
  {
//--- coordonnées
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+(BUTTON_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- création
   if(!m_spin_edit.Create(m_chart_id,m_name+"SpinEdit",m_subwin,x1,y1,x2,y2))
      return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!Add(m_spin_edit))
      return(false);
   m_spin_edit.MinValue(10);
   m_spin_edit.MaxValue(100);
   m_spin_edit.Value(50);
   Comment(__FUNCTION__+" : Valeur="+IntegerToString(m_spin_edit.Value()));
//--- succès
   return(true);
  }
//+------------------------------------------------------------------+
//| Gestionnaire d'évènement                                         |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeSpinEdit(void)
  {
   Comment(__FUNCTION__+" : Valeur="+IntegerToString(m_spin_edit.Value()));
  }
//+------------------------------------------------------------------+
//| Variables Globales                                               |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Fonction d'initialisation de l'Expert       |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crée le dialogue de l'application
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- lance l'application
   ExtDialog.Run();
//--- succès
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Fonction de désinitialisation de l'Expert                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprime le commentaire
   Comment("");
//--- détruit le dialogue
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Fonction d'évènement du graphique                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // identifiant de l'évènement
                  const long& lparam,    // paramètre d'évènement de type long
                  const double& dparam,  // paramètre d'évènement de type double
```

© 2000-2024, MetaQuotes Ltd.

```
                    const string& sparam) // paramètre d'évènement de type string
{
 ExtDialog.ChartEvent(id,lparam,dparam,sparam);
}
```

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CSpinEdit.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

MinValue (Méthode "Get")

Retourne la valeur de la propriété "MinValue" (valeur minimale permise) du contrôle.

```
int  MinValue()  const
```

Valeur de retour

La valeur de la propriété "MinValue".

MinValue (Méthode "Set")

Définit la valeur de la propriété "MinValue" (valeur minmale permise) du contrôle.

```
void  MinValue(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "MinValue".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

MaxValue (Méthode "Get")

Retourne la valeur de la propriété "MaxValue" (valeur maximale permise) du contrôle.

```
int  MaxValue()  const
```

Valeur de retour

La valeur de la propriété "MaxValue".

MaxValue (Méthode "Set")

Définit la valeur de la propriété "MaxValue" (valeur maximale permise) du contrôle.

```
void  MaxValue(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "MaxValue".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Value (Méthode "Get")

Retourne la valeur de la propriété "Value" (valeur actuelle) du contrôle.

```
int  Value()  const
```

Valeur de retour

La valeur de la propriété "Value".

Value (Méthode "Set")

Définit la valeur de la propriété "Value" (valeur actuelle) du contrôle.

```
void  Value(
   const int  value      // valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "Value".

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

CreateEdit

Crée un contrôle dépendant (CEdit).

```
virtual bool  CreateEdit()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateInc

Crée un contrôle dépendant (bouton haut)

```
virtual bool  CreateInc()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateDec

Crée un contrôle dépendant (bouton bas).

```
virtual bool  CreateDec()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnClickInc

Le gestionnaire d'évènement "ClickInc" (clic de la souris sur le bouton d'incrémentation) du contrôle.

```
virtual bool  OnClickInc()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnClickDec

Le gestionnaire d'évènement "ClickDec" (clic de la souris sur le bouton de décrémentation) du contrôle.

```
virtual bool  OnClickDec()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

OnChangeValue

Le gestionnaire d'évènement "ChangeValue" du contrôle.

```
virtual bool  OnChangeValue()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2024, MetaQuotes Ltd.

CDialog

La classe CDialog est une classe de contrôle complexe de Dialog.

Description

La classe CAppDialog permet de combiner des contrôles ayant des fonctions différentes dans un groupe au sein d'un groupe.

Déclaration

```
class CDialog : public CWndContainer
```

Titre

```
#include <Controls\Dialog.mqh>
```

Hiérarchie d'héritage

COBJECT
CWnd
CWndContainer
CDialog

Descendants directs

CAppDialog

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
Caption	Retourne/Définit la valeur de la propriété "Caption"
Add	
Add	Ajoute un contrôle dans la zone client
Contrôles dépendants	
CreateWhiteBorder	Crée un contrôle dépendant (bordure blanche)
CreateBackground	Crée un contrôle dépendant (arrière plan)
CreateCaption	Crée un contrôle dépendant (légende)

© 2000-2024, MetaQuotes Ltd.

Création	
CreateButtonClose	Crée un contrôle dépendant (bouton Fermer)
CreateClientArea	Crée un contrôle dépendant (zone client)
Gestionnaires d'évènements des contrôles dépendants	
OnClickCaption	Gestionnaire d'évènement "ClickCaption"
OnClickButtonClose	Gestionnaire d'évènement "ClickButtonClose"
Accés à la zone client	
ClientAreaVisible	Définit une valeur indiquant si la zone client est visible ou pas
ClientAreaLeft	Retourne la coordonnée X du coin supérieur gauche de la zone client du contrôle
ClientAreaTop	Retourne la coordonnée Y du coin supérieur gauche de la zone client du contrôle
ClientAreaRight	Retourne la coordonnée X du coin inférieur droit de la zone client du contrôle
ClientAreaBottom	Retourne la coordonnée Y du coin inférieur droit de la zone client du contrôle
ClientAreaWidth	Retourne la largeur de la zone client
ClientAreaHeight	Retourne la hauteur de la zone client
Gestionnaires d'évènement de Drag n' Drop	
OnDialogDragStart	Gestionnaire (virtuel) de l'évènement "DialogDragStart"
OnDialogDragProcess	Gestionnaire (virtuel) de l'évènement "DialogDragProcess"
OnDialogDragEnd	Gestionnaire (virtuel) de l'évènement "DialogDragEnd"

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CDialog.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

Caption (Méthode "Get")

Retourne la propriété "Caption" du contrôle CDialog.

```
string  MinValue()  const
```

Valeur de retour

La propriété "Caption".

Caption (Méthode "Set")

Définit la propriété "Caption" du contrôle CDialog.

```
bool  Caption(
   const string  text      // texte
   )
```

Paramètres

text

[in] Nouvelle valeur de la propriété "Caption".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Add

Ajoute le contrôle dans la zone client par pointeur.

```
bool  Add(
   CWnd   *control,         // pointeur
   )
```

Paramètres

control

[in] Pointeur vers le contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

Add

Ajoute le contrôle dans la zone client par référence.

```
bool  Add(
   CWnd   &control,         // référence
   )
```

Paramètres

control

[in] Référence sur le contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateWhiteBorder

Crée un contrôle dépendant (bordure blanche).

```
virtual bool  CreateWhiteBorder()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateBackground

Crée un contrôle dépendant (arrière plan).

```
virtual bool  CreateBackground()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateCaption

Crée un contrôle dépendant (légende).

```
virtual bool  CreateCaption()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateButtonClose

Crée un contrôle dépendant (bouton Fermer)

```
virtual bool  CreateButtonClose()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateClientArea

Crée un contrôle dépendant (zone client)

```
virtual bool  CreateClientArea()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnClickCaption

Le gestionnaire d'évènement "ClickCaption" du contrôle.

```
virtual bool  OnClickCaption()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

OnClickButtonClose

Le gestionnaire d'évènement "ClickButtonClose" du contrôle.

```
virtual bool  OnClickButtonClose()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ClientAreaVisible

Définit une valeur indiquant si la zone client est visible ou pas.

```
bool  ClientAreaVisible(
   const bool  visible      // flag de visibilité
   )
```

Paramètres

visible

[in] Flag de visibilité.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ClientAreaLeft

Retourne la coordonnée X du coin supérieur gauche de la zone client du contrôle.

```
int  ClientAreaLeft()
```

Valeur de retour

La coordonnée X du coin supérieur gauche de la zone client du contrôle.

© 2000-2024, MetaQuotes Ltd.

ClientAreaTop

Retourne la coordonnée Y du coin supérieur gauche de la zone client du contrôle.

```
int  ClientAreaTop()
```

Valeur de retour

La coordonnée Y du coin supérieur gauche de la zone client du contrôle.

© 2000-2024, MetaQuotes Ltd.

ClientAreaRight

Retourne la coordonnée X du coin inférieur droit de la zone client du contrôle.

```
int  ClientAreaTop()
```

Valeur de retour

La coordonnée X du coin inférieur droit de la zone client du contrôle.

© 2000-2024, MetaQuotes Ltd.

ClientAreaBottom

Retourne la coordonnée Y du coin inférieur droit de la zone client du contrôle.

```
int  ClientAreaBottom()
```

Valeur de retour

La coordonnée Y du coin inférieur droit de la zone client du contrôle.

© 2000-2024, MetaQuotes Ltd.

ClientAreaWidth

Retourne la largeur de la zone client du contrôle.

```
int  ClientAreaWidth()
```

Valeur de retour

La largeur de la zone client du contrôle.

© 2000-2024, MetaQuotes Ltd.

ClientAreaHeight

Retourne la hauteur de la zone client du contrôle.

```
int  ClientAreaHeight()
```

Valeur de retour

La hauteur de la zone client du contrôle.

© 2000-2024, MetaQuotes Ltd.

OnDialogDragStart

Le gestionnaire d'évènement "DialogDragStart" du contrôle.

```
virtual bool  OnDialogDragStart()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DialogDragStart" est généré au démarrage du déplacement du contrôle (par Drag n' Drop).

© 2000-2024, MetaQuotes Ltd.

OnDialogDragProcess

Le gestionnaire d'évènement "DialogDragProcess" du contrôle.

```
virtual bool  OnDialogDragProcess()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DialogDragProcess" est généré lorsque le contrôle est déplacé (par Drag n' Drop).

© 2000-2024, MetaQuotes Ltd.

OnDialogDragEnd

Le gestionnaire d'évènement "DialogDragEnd" du contrôle.

```
virtual bool  OnDialogDragEnd()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DialogDragEnd" est généré à la fin du déplacement du contrôle (par Drag n' Drop).

© 2000-2024, MetaQuotes Ltd.

CAppDialog

La classe CAppDialog est une classe de contrôle complexe de Dialogue d'Application (avec des contrôles inter-dépendants).

Description

La classe CAppDialog permet de combiner des contrôles ayant des fonctions différentes dans un groupe au sein d'un programme MQL5.

Déclaration

```
class CAppDialog : public CDialog
```

Titre

```
#include <Controls\Dialog.mqh>
```

Hiérarchie d'héritage

CObject
  CWnd
    CWndContainer
      CDialog
        CAppDialog

Méthodes de Classe

Création et destruction	
Create	Crée le contrôle
Destroy	Détruit le contrôle
Traitement des évènements	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Exécution	
Run	Exécute le contrôle
Traitement des évènements graphiques	
ChartEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Paramètres	
Minimized	Définit une valeur indiquant si le contrôle est minimisé
Sauvegarde/Chargement	

© 2000-2024, MetaQuotes Ltd.

Création et destruction	
IniFileSave	Sauvegarde l'état du contrôle dans un fichier
IniFileLoad	Charge l'état du contrôle depuis un fichier
IniFileName	Définit le nom du fichier de chargement/sauvegarde de l'état du contrôle
IniFileExt	Définit l'extension du nom de fichier de chargement/sauvegarde de l'état du contrôle
Initialisation	
CreateCommon	Méthode standard d'initialisation
CreateExpert	Méthode d'initialisation pour travailler dans un Expert Advisor
CreateIndicator	Méthode d'initialisation pour travailler dans un indicateur
Contrôles dépendants	
CreateButtonMinMax	Crée des des contrôles dépendants (boutons minimiser et maximiser)
Gestionnaires d'évènements des contrôles dépendants	
OnClickButtonClose	Gestionnaire (virtuel) de l'évènement "ClickButtonClose"
OnClickButtonMinMax	Gestionnaire (virtuel) de l'évènement "ClickButtonMinMax"
Evènements externes	
OnAnotherApplicationClose	Gestionnaire (virtuel) d'évènements externes
Méthodes	
Rebound	Définit les nouvelles coordonnées du contrôle en utilisant les coordonnées de la classe CRect
Minimize	Réduit le contrôle
Maximize	Agrandit le contrôle à son état précédent
CreateInstanceId	Crée un identifiant unique pour les noms des objets de contrôle
ProgramName	Retourne le nom du programme MQL5
SubwinOff	Retourne le décalage Y de la sous-fenêtre du contrôle

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

Méthodes héritées de la classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet,

© 2000-2024, MetaQuotes Ltd.

Méthodes héritées de la classe CObject

Prev, Prev, Next, Next, Type, Compare

StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Méthodes héritées de la classe CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Méthodes héritées de la classe CDialog

Caption, Caption, Add, Add

© 2000-2024, MetaQuotes Ltd.

Create

Crée un nouveau contrôle CAppDialog.

```
virtual bool  Create(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Destroy

Méthode de désinitialisation du contrôle CAppDialog.

```
virtual void  Destroy(
   const int  reason=REASON_PROGRAM    // code de la raison
)
```

Paramètres

reason

[in] Code de la raison de la désinitialisation. REASON_PROGRAM est utilisé par défaut.

Valeur de Retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

Run

Exécute le contrôle.

```
bool  Run()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

ChartEvent

Gestionnaire d'évènement graphique.

```
virtual bool  ChartEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2024, MetaQuotes Ltd.

Minimized

Définit la valeur de la propriété "Minimized" du contrôle.

```
bool  Minimized(
   const bool  flag      // état
   )
```

Paramètres

flag

[in] Nouvel état.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

IniFileSave

Sauvegarde l'état du contrôle dans un fichier

```
void  IniFileSave()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

IniFileLoad

Charge l'état du contrôle depuis un fichier

```
void  IniFileLoad()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

IniFileName

Définit le nom du fichier de chargement/sauvegarde de l'état du contrôle

```
virtual string  IniFileName()  const
```

Valeur de retour

Nom du fichier de chargement/sauvegarde de l'état du contrôle.

Note

Le nom du fichier inclut le nom de l'Expert Advisor/indicateur et le nom du symbole, sur lequel le programme MQL5 est lancé.

© 2000-2024, MetaQuotes Ltd.

IniFileExt

Définit l'extension du nom de fichier de chargement/sauvegarde de l'état du contrôle

```
virtual string  IniFileExt()  const
```

Valeur de retour

Extension du fichier, utilisé pour le chargement/la sauvegarde de l'état du contrôle.

© 2000-2024, MetaQuotes Ltd.

CreateCommon

Méthode standard d'initialisation.

```
bool  CreateCommon(
   const long    chart,       // identifiant du graphique
   const string  name,        // nom
   const int     subwin,      // sous-fenêtre du graphique
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateExpert

Méthode d'initialisation pour travailler dans un Expert Advisor

```
bool  CreateExpert(
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateIndicator

Méthode d'initialisation pour travailler dans un indicateur

```
bool  CreateIndicator(
   const int     x1,          // coordonnée x1
   const int     y1,          // coordonnée y1
   const int     x2,          // coordonnée x2
   const int     y2           // coordonnée y2
   )
```

Paramètres

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateButtonMinMax

Crée des contrôles inter-dépendants (boutons minimiser/maximiser).

```
virtual void  CreateButtonMinMax()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnClickButtonClose

Le gestionnaire d'évènement "ClickButtonClose" (clic de souris sur le bouton Fermer) du contrôle.

```
virtual void  OnClickButtonClose()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnClickButtonMinMax

Le gestionnaire d'évènement "ClickButtonMinMax" (clic de souris sur l'un des boutons minimiser/maximiser) du contrôle.

```
virtual void  OnClickButtonClose()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

OnAnotherApplicationClose

Gestionnaire d'évènement des évènements externes.

```
virtual void  OnAnotherApplicationClose()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Rebound

Définit les nouvelles coordonnées du contrôle utilisant les coordonnées de la classe CRect.

```
bool  Rebound(
   const & CRect  rect      // Classe CRect
   )
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Minimize

Réduit le contrôle

```
virtual void  Minimize()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

Maximize

Agrandit le contrôle à son état précédent.

```
virtual void  Maximize()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2024, MetaQuotes Ltd.

CreateInstanceId

Crée un identifiant unique pour les noms des objets de contrôle.

```
string  CreateInstanceId()
```

Valeur de retour

Préfixe à utiliser pour les noms des objets.

© 2000-2024, MetaQuotes Ltd.

ProgramName

Retourne le nom du programme MQL5.

```
string  ProgramName()
```

Valeur de retour

Nom du programme MQL5.

© 2000-2024, MetaQuotes Ltd.

SubwinOff

Retourne le décalage Y de la sous-fenêtre du contrôle.

```
void  SubwinOff()
```

Valeur de retour

Aucune.

© 2000-2024, MetaQuotes Ltd.

Le passage de MQL4

Le langage MQL5 est le développement du prédécesseur – le langage MQL4, dans laquel de nombreux indicateurs, des scripts et des experts ont été écrits. Malgré que le langage moderne de programmation est au maximum compatible avec le langage de la génération précédente, néanmoins il y a une série de différences entre ces langages. Et il faut connaître ces différences au transfert des programmes.

Dans ce paragraphe on a rassemblé l'information, qui est appelée à faciliter l'adaptation de ses codes à ces programmeurs qui savent bien MQL4, au langage nouveau MQL5.

Tout d'abord il est nécessaire de noter:

- Les fonctions start(), init() et deinit() manquent;
- Le nombre de tampons d'indicateur n'est pas limitée;
- Le chargement dll se passe à la fois après le chargement de l'expert (ou n'importe quel autre programme mql5);
- La vérification raccourcie des conditions logiques;
- À la sortie au-delà du tableau l'exécution courante se cesse (critiquement - avec la sortie de l'erreur);
- La priorité des opérations comme dans C++;
- La réduction implicite des types travaille (même de la chaîne dans le nombre);
- Les variables locales ne sont pas initialisées automatiquement (excepté les chaînes);
- Les tableaux ordinaires locaux sont supprimés automatiquement.

Les fonctions spéciales init, start et deinit

Dans le langage MQL4 était seulement trois fonctions prédéterminées, qui pouvaient être dans le code de l'indicateur, le conseiller ou le script (sans prendre en compte les fichiers inclus *.mqh et les fichiers des bibliothèques). Ces fonctions sont absentes à MQL5, mais il y a leurs analogues. Dans le tableau on présente la conformité approximative des fonctions.

MQL4	MQL5
init	OnInit
start	OnStart
deinit	OnDeinit

Les fonctions OnInit et OnDeinit jouent le même rôle que les fonctions init et deinit dans MQL4 – elles se sont destinées au placement du code, qui doit être accompli à l'initialisation et la déinitialisation des programmes mql5. Vous pouvez tout simplement rebaptiser ces fonctions de la bonne façon, ou les laisser comme elles sont, mais ajouter l'appel de ces fonctions dans les espaces correspondants.

Exemple:

© 2000-2024, MetaQuotes Ltd.

```
void OnInit()
  {
//--- appelons la fonction à l'initialisation
   init();
  }
void OnDeinit(const int reason)
  {
//--- appelons la fonction à la déinitialisation
   deinit();
//---
  }
```

La fonction start est remplacée sur OnStart seulement dans les scripts, mais pour l'expert et l'indicateur il est nécessaire de la rebaptiser conformément à OnTick et OnCalculate. Notamment dans ces trois fonctions il est nécessaire d'installer le code, qui doit être accompli au temps de travail du programme mql5:

programme mql5	fonction principale
le script	OnStart
l'indicateur	OnCalculate
l'expert	OnTick

Si la fonction principale manque dans le code exécuté de l'indicateur ou le script, ou le nom de cette fonction se distingue de demandé, l'appel d'une telle fonction n'est pas produit. C'est-à-dire, si dans le code source du script il n'y a pas de fonction OnStart, un tel code sera compilé comme le conseiller.

Si dans le code de l'indicateur il n'y a pas de fonction OnCalculate, la compilation d'un tel indicateur est impossible.

Les variables prédéterminées

A MQL5 il n'y a pas de telles variables prédéterminées comme Ask, Bid, Bars. Les variables Digits et Point ont changé un peu dans l'orthographe, comme c'est montré dans le tableau.

MQL4	MQL5
Digits	_Digits
Point	_Point
	_LastError
	_Period
	_Symbol
	_StopFlag
	_UninitReason

L'accès aux séries temporelles

© 2000-2024, MetaQuotes Ltd.

A MQL5 il n'y a pas des séries temporelles prédéterminées Open[], High[], Low[], Close[], Volume[] et Time[]. On peut spécifier la profondeur nécessaire de la série temporelle indépendamment à l'aide des fonctions correspondantes pour l'accès aux séries temporelles.

Les conseillers (les experts)

Les experts dans MQL5 doivent avoir pas absolument la fonction-gestionnaire de l'événement de l'entrée du nouveau tick OnTick, comme c'était à MQL4 (la fonction start dans MQL4 est appellée pour l'exécution à l'entrée du nouveau tick), parce que maintenant les experts dans MQL5 peuvent contenir les fonctions-gestionnaire de quelques types des événements:

- OnTick – L'entrée du nouveau tick;
- OnTimer –l'événement du minuteur;
- OnTrade - l'événement commercial;
- OnChartEvent – Les événements de l'entrée du clavier et de la souris, les événements de la navigation de l'objet graphique, l'événement de la fin de l'édition du texte dans la zone de l'entréé de l'objet LabelEdit;
- OnBookEvent – l'événement du changement de l'état du profondeur de marché (Depth of Market).

Les indicateurs d'utilisateur

A MQL4 le nombre de tampons d'indicateur est limitée et ne peut pas être plus que 8. Il n'y a pas de tel bornage dans MQL5, mais il faut se rappeler que chaque tampon d'indicateur demande la sélection du volume défini de la mémoire vive sous son placement dans le terminal, c'est pourquoi il ne faut pas abuser d'une nouvelle possibilité ouverte quand même.

En outre, dans MQL4 il y avait seulement 6 types du dessin de l'indicateur d'utilisateur, dans MQL5 maintenant il y a 18 styles du dessin. Et bien que les noms des aspects du dessin n'ont pas changé, l'idéologie de l'image graphique des indicateurs a changé beaucoup.

Se distingue également la direction de l'indexation dans les tampons d'indicateur. Dans MQL5 tous les tampons se comportent par défaut comme les tableaux ordinaires, c'est-à-dire l'élément avec l'index 0 est plus vieil à l'histoire, à l'augmentation de l'index nous nous avançons des données les plus vieilles vers les données dernières.

La seule fonction du travail avec les indicateurs d'utilisateur, qui a été préservé de MQL4 – c'est SetIndexBuffer. Mais son appel a changé aussi, maintenant il faut indiquer le type de données, qui se trouveront dans le tableau , lié au tampon d'indicateur.

En outre ont changé et se sont élargies les propriétés des indicateurs d'utilisateur, il y avait des fonctions de l'accès aux séries temporelles, c'est pourquoi il est nécessaire entièrement de repenser tout l'algorithme de calcul.

Les objets graphiques

Le nombre d'objets graphiques à MQL5 s'est élargie beaucoup. En outre le positionnement des objets graphiques dans le temps est devenu possible avec l'exactitude d'une seconde dans un graphique de n'importe quelle période – maintenant on ne produit pas l'arrondissement des points d'attachement des objets graphiques avec l'exactitude jusqu'au temps de l'ouverture de la barre sur le graphique du prix courant.

© 2000-2024, MetaQuotes Ltd.

Pour les objets Arrow, Text et Label il y avait une possibilité d'indiquer le moyen du rattachement, et pour les objets Label, Button, Chart, Bitmap Label et Edit il y a la possibilité de spécifier l'angle du graphique, auquel l'objet est attaché.

© 2000-2024, MetaQuotes Ltd.

List of MQL5 Functions

All MQL5 functions in alphabetical order.

Fonction	Action	Раздел
AccountInfoDouble	Rend la valeur du type double de la propriété correspondante du compte	Information sur le compte
AccountInfoInteger	Rend la valeur du type entier (bool,int ou long) de la propriété correspondante du compte	Information sur le compte
AccountInfoString	Rend la valeur du type string de la propriété correspondante du compte	Information sur le compte
acos	Rend la valeur d'arc cosinus x aux radians	Fonctions mathématiques
Alert	Affiche le message dans la fenêtre séparée	Fonctions communes
ArrayBsearch	Rend l'index du premier élément trouvé dans dans la première dimension du tableau	Opérations avec les tableaux
ArrayCompare	Rend le résultat de la comparaison de deux tableux des types simples ou des structures d'utilisateur qui n'ont pas des objets complexes	Opérations avec les tableaux
ArrayCopy	Copie un tableau à l'autre	Opérations avec les tableaux
ArrayFill	Remplit le tableau numérique par la valeur indiquée	Opérations avec les tableaux
ArrayFree	Libère la mémoire du tampon de n'importe quel tableau dynamique et met la dimension de la valeur zéro au 0.	Opérations avec les tableaux
ArrayGetAsSeries	Contrôle la direction de l'indexation du tableau	Opérations avec les tableaux
ArrayInitialize	Met tous les éléments du tableau numérique à une valeur	Opérations avec les tableaux
ArrayIsDynamic	Vérifies - si le tableau est dynamique	Opérations avec les tableaux
ArrayIsSeries	Vérifies - si le tableau est la série temporelle	Opérations avec les tableaux

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
ArrayMaximum	La recherche de l'élément avec la valeur maximal	Opérations avec les tableaux
ArrayMinimum	La recherche de l'élément avec la valeur minimal	Opérations avec les tableaux
ArrayRange	Rend le nombre d'éléments dans la dimension indiquée du tableau	Opérations avec les tableaux
ArrayResize	Met la nouvelle grandeur dans la première dimension du tableau	Opérations avec les tableaux
ArraySetAsSeries	Établit la direction de l'indexation dans le tableau	Opérations avec les tableaux
ArraySize	Rend le nombre d'éléments dans le tableau	Opérations avec les tableaux
ArrayPrint	Ecrit un tableau de types simples ou une structure simple dans le journal	Fonctions sur les Tableaux
ArrayInsert	Insère le nombre spécifié d'éléments d'un tableau source dans un tableau destination à partir d'un indice spécifié	Fonctions sur les Tableaux
ArrayRemove	Supprime le nombre spécifié d'éléments du tableau à partir de l'indice spécifié	Fonctions sur les Tableaux
ArrayReverse	Inverse le nombre spécifié d'éléments dans le tableau en commencant à l'indice spécifié	Fonctions sur les Tableaux
ArraySwap	Intervertit les contenus de deux tableaux dynamiques du même type	Fonctions sur les Tableaux
ArraySort	Le triage des tableaux numériques par la première dimension	Opérations avec les tableaux
asin	Rend la valeur d'arc sinus x aux radians	Fonctions mathématiques
atan	Rend l'arc tangent x aux radians	Fonctions mathématiques
Barres	Rend le nombre de barres aux histoires selon le symbole correspondant et la période	Accès aux timeséries et les données des indicateurs
BarsCalculated	Rend la quantité de données calculées dans le tampon	Accès aux timeséries et les données des indicateurs

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	d'indicateur ou-1 en cas de l'erreur (les données ne sont pas encore calculées)	
CalendarCountryById	Retourne la description d'un pays par son identifiant	Calendrier Economique
CalendarEventById	Retourne la description d'un évènement par son identifiant	Calendrier Economique
CalendarValueById	Retourne la description de la valeur d'un évènement par son identifiant	Calendrier Economique
CalendarCountries	Retourne le tableau des noms des pays disponibles dans le calendrier	Calendrier Economique
CalendarEventByCountry	Retourne le tableau des descriptions de tous les évènements disponibles dans le calendrier pour un code de pays spécifié	Calendrier Economique
CalendarEventByCurrency	Retourne le tableau des descriptions de tous les évènements disponibles dans le calendrier pour une devise spécifiée	Calendrier Economique
CalendarValueHistoryByEvent	Retourne le tableau des valeurs pour tous les évènements dans un intervalle de temps spécifié pour un identifiant d'évènement	Calendrier Economique
CalendarValueHistory	Retourne le tableau des valeurs pour tous les évènements dans un intervalle de temps spécifié avec la possibilité de trier par pays et/ou devise	Calendrier Economique
CalendarValueLastByEvent	Retourne le tableau des valeurs d'un évènement par son identifiant depuis le statut de la base de données du calendrier avec un change_id spécifié	Calendrier Economique
CalendarValueLast	Retourne le tableau des valeurs de tous les évènements, avec la possibilité de trier par pays et/devise depuis le statut de la base de données du calendrier avec un change_id spécifié	Calendrier Economique

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
ceil	Rend la valeur entière numérique la plus proche par dessus	Fonctions mathématiques
CharArrayToString	Convertit le code de symbole (ansi) à la chaîne d'un symbole	Conversion des données
ChartApplyTemplate	Applique au graphique indiqué le modèle du fichier indiqué	Opérations avec les graphiques
ChartClose	Ferme le graphique indiqué	Opérations avec les graphiques
ChartFirst	Rend l'identificateur du premier graphique du terminal de client	Opérations avec les graphiques
ChartGetDouble	Rend la valeur de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartGetInteger	Rend la valeur entière de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartGetString	Rend la valeur de chaîne de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartID	Rend l'identificateur du graphique courant	Opérations avec les graphiques
ChartIndicatorAdd	Ajoute l'indicateur avec le handle indiqué sur la fenêtre indiquée du graphique	Opérations avec les graphiques
ChartIndicatorDelete	Supprime l'indicateur avec le nom spécifié de la fenêtre indiquée du graphique	Opérations avec les graphiques
ChartIndicatorGet	Rend le handle de l'indicateur avec le nom court indiqué sur la fenêtre indiquée du graphique	Opérations avec les graphiques
ChartIndicatorName	Rend le nom court de l'indicateur selon le numéro dans la liste des indicateurs sur la fenêtre indiquée du graphique	Opérations avec les graphiques
ChartIndicatorsTotal	Rend le nombre de tous indicateurs qui sont attachés à la fenêtre indiquée du graphique	Opérations avec les graphiques
ChartNavigate	Réalise le décalage du graphique indiqué au nombre indiqué de	Opérations avec les graphiques

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	barres par rapport à la position indiquée du graphique	
ChartNext	Rend l'identificateur du graphique, suivant de l'indiqué	Opérations avec les graphiques
ChartOpen	Ouvre un nouveau graphique avec le symbole indiqué et la période	Opérations avec les graphiques
CharToString	La conversion du code de symbole dans une chaîne d'un symbole	Conversion des données
ChartPeriod	Rend la valeur de la période du graphique indiqué	Opérations avec les graphiques
ChartPriceOnDropped	Rend la coordonnée de prix correspondant au point, dans laquelle on jette par la souris l'expert donné ou le script	Opérations avec les graphiques
ChartRedraw	Appelle la copie forcée du graphique indiqué	Opérations avec les graphiques
ChartSaveTemplate	Enregistre les paramètres actuels du graphique dans un modèle avec le nom spécifié	Opérations avec les graphiques
ChartScreenShot	Fournit un screenshot du graphique de son état actuel dans un format de GIF, PNG ou BMP en fonction de l'extension spécifiée	Opérations avec les graphiques
ChartSetDouble	Spécifie la valeur du type double de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartSetInteger	Spécifie la valeur du type entier (datetime, int, color, bool ou char) de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartSetString	Spécifie la valeur du type string de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartSetSymbolPeriod	Change les valeurs du symbole et la période du graphique indiqué	Opérations avec les graphiques
ChartSymbol	Rend le nom du symbole du graphique indiqué	Opérations avec les graphiques
ChartTimeOnDropped	Rend la coordonnée temporaire correspondant au point, dans	Opérations avec les graphiques

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	laquelle on jette par la souris l'expert donné ou le script	
ChartTimePriceToXY	Convertit les coordonnées du graphique de la représentation graphique du temps/prix pour les axes X et Y	Opérations avec les graphiques
ChartWindowFind	Rend le numéro de sous- fenêtre, dans laquelle il y a un indicateur	Opérations avec les graphiques
ChartWindowOnDropped	Rend le numéro de sous-fenêtre du graphique, sur laquelle on jette l'expert donné, le script, l'objet ou l'indicateur par la souris	Opérations avec les graphiques
ChartXOnDropped	Rend la coordonnée selon l'axe X, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris	Opérations avec les graphiques
ChartXYToTimePrice	Convertit les coordonnées X et Y du graphique en valeurs - le temps et le prix	Opérations avec les graphiques
ChartYOnDropped	Rend la coordonnée selon l'axe Y, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris	Opérations avec les graphiques
CheckPointer	Rend le type de l'indication de l'objet	Fonctions communes
CLBufferCreate	Crée le tampon OpenCL	Travail avec OpenCL
CLBufferFree	Supprime le tampon OpenCL	Travail avec OpenCL
CLBufferRead	Lit le tampon OpenCL au tableau	Travail avec OpenCL
CLBufferWrite	Inscrit le tableau au tampon OpenCL	Travail avec OpenCL
CLContextCreate	Crée le contexte OpenCL	Travail avec OpenCL
CLContextFree	Supprime le contexte OpenCL	Travail avec OpenCL
CLExecute	Exécute le programme OpenCL	Travail avec OpenCL
CLGetDeviceInfo	Reçoit la propriété du dispositif du pilote informatique OpenCL	Travail avec OpenCL
CLGetInfoInteger	Rend la valeur de la propriété entière pour l'objet OpenCL ou le dispositif	Travail avec OpenCL

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
CLHandleType	Rend le type OpenCL du handle à titre de la valeur de l'énumération ENUM_OPENCL_HANDLE_TYPE	Travail avec OpenCL
CLKernelCreate	Crée la fonction du lancement de OpenCL	Travail avec OpenCL
CLKernelFree	Supprime la fonction du lancement de OpenCL	Travail avec OpenCL
CLProgramCreate	Crée le programme OpenCL du code initial	Travail avec OpenCL
CLProgramFree	Supprime le programme OpenCL	Travail avec OpenCL
CLSetKernelArg	Expose le paramètre de la fonction OpenCL	Travail avec OpenCL
CLSetKernelArgMem	Expose le tampon OpenCL comme un paramètre de la fonction OpenCL	Travail avec OpenCL
ColorToARGB	Convertit le type color en type uint pour recevoir la représentation ARGB de la couleur.	Conversion des données
ColorToString	Convertit la valeur de la couleur à la chaîne du type "R,G,B"	Conversion des données
Comment	Affiche le message à un angle gauche supérieur du graphique de prix	Fonctions communes
CopyBuffer	Reçoit au tableau les données du tampon indiqué de l'indicateur indiqué	Accès aux timeséries et les données des indicateurs
CopyClose	Reçoit au tableau les données historiques au prix de la clôture des barres selon le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopyHigh	Reçoit au tableau les données historiques selon le prix maximum des barres selon le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopyLow	Reçoit au tableau les données historiques selon le prix minimum des barres selon le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
CopyOpen	Reçoit au tableau les données historiques au prix de l'ouverture des barres selon le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopyRates	Reçoit au tableau les données historiques de la structure Rates pour le symbole et la période indiqués	Accès aux timeséries et les données des indicateurs
CopyRealVolume	Reçoit au tableau les données historiques par les volumes commerciaux pour le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopySpread	Reçoit au tableau les données historiques selon les spread pour le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopyTicks	Получает в массив тики, накопленные терминалом за текущую рабочую сессию	Accès aux timeséries et les données des indicateurs
CopyTickVolume	Reçoit au tableau les données historiques par les volumes des ticks pour le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopyTime	Reçoit au tableau les données historiques par le temps de l'ouverture des barres selon le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
cos	Rend le cosinus du nombre	Fonctions mathématiques
CryptDecode	Производит обратное преобразование данных массива	Fonctions communes
CryptEncode	Преобразует данные массива-источника в массив-приемник указанным методом	Fonctions communes
CustomSymbolCreate	Crée un symbole personnalisé avec le nom spécifié dans le groupe spécifié	Symboles personnalisés
CustomSymbolDelete	Supprime un symbole personnalisé ayant le nom spécifié	Symboles personnalisés
CustomSymbolSetInteger	Assigne une valeur de type integer à la propriété d'un	Symboles personnalisés

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	symbole personnalisé	
CustomSymbolSetDouble	Assigne une valeur de type réel à la propriété d'un symbole personnalisé	Symboles personnalisés
CustomSymbolSetString	Assigne une valeur de type string à la propriété d'un symbole personnalisé	Symboles personnalisés
CustomSymbolSetMarginRate	Définit les taux de marge suivant le type et la direction de l'ordre pour un symbole personnalisé	Symboles personnalisés
CustomSymbolSetSessionQuote	Définit les heures de début et de fin de la session de cotation spécifiée pour le symbole et le jour de la semaine spécifiés	Symboles personnalisés
CustomSymbolSetSessionTrade	Définit les heures de début et de fin de la session de cotation spécifiée pour le symbole et le jour de la semaine spécifiés	Symboles personnalisés
CustomRatesDelete	Supprime toutes les barres de l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié	Symboles personnalisés
CustomRatesReplace	Remplace tout l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié avec les données du tableau de données de type MqlRates	Symboles personnalisés
CustomRatesUpdate	Ajoute des barres manquantes dans l'historique du symbole personnalisé et remplace les données existantes avec celles du tableau de données de type MqlRates	Symboles personnalisés
CustomTicksAdd	Adds data from an array of the MqlTick type to the price history of a custom symbol. The custom symbol must be selected in the Market Watch window	Symboles personnalisés
CustomTicksDelete	Supprime tous les ticks de l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié	Symboles personnalisés

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
CustomTicksReplace	Remplace tout l'historique des prix du symbole personnalisé dans l'intervalle de temps spécifié avec les données du tableau de données de type MqlTick	Symboles personnalisés
CustomBookAdd	Passe le statut du Depth of Market pour un symbole personnalisé	Symboles personnalisés
DatabaseOpen	Ouvre ou crée une base de données dans le fichier spécifié	Utilisation de bases de données
DatabaseClose	Ferme une base de données	Utilisation de bases de données
DatabaseImport	Imports data from a file into a table	Utilisation de bases de données
DatabaseExport	Exports a table or an SQL request execution result to a CSV file	Utilisation de bases de données
DatabasePrint	Prints a table or an SQL request execution result in the Experts journal	Utilisation de bases de données
DatabaseTableExists	Vérifie la présence d'une table dans la base de données	Utilisation de bases de données
DatabaseExecute	Exécute une requête dans la base de données spécifiée	Utilisation de bases de données
DatabasePrepare	Crée le handle d'une requête, qui peut être ensuite exécutée avec DatabaseRead()	Utilisation de bases de données
DatabaseReset	Réinitialise une requête, comme après avoir appelé DatabasePrepare()	Utilisation de bases de données
DatabaseBind	Définit la valeur d'un paramètre dans une requête	Utilisation de bases de données
DatabaseBindArray	Définit un tableau comme valeur de paramètre	Utilisation de bases de données
DatabaseRead	Se déplace sur l'enregistrement suivant des résultats de la requête	Utilisation de bases de données
DatabaseReadBind	Moves to the next record and reads data into the structure from it	Utilisation de bases de données

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
DatabaseFinalize	Supprime une requête créée avec DatabasePrepare()	Utilisation de bases de données
DatabaseTransactionBegin	Démarre l'exécution d'une transaction	Utilisation de bases de données
DatabaseTransactionCommit	Termine l'exécution d'une transaction	Utilisation de bases de données
DatabaseTransactionRollback	Annule une transaction	Utilisation de bases de données
DatabaseColumnsCount	Retourne le nombre de colonnes dans une requête	Utilisation de bases de données
DatabaseColumnName	Retourne le nom d'une colonne par son numéro	Utilisation de bases de données
DatabaseColumnType	Retourne le type d'une colonne par son numéro	Utilisation de bases de données
DatabaseColumnSize	Retourne la taille d'une colonne en octets	Utilisation de bases de données
DatabaseColumnText	Retourne la valeur d'une colonne de type string de l'enregistrement courant	Utilisation de bases de données
DatabaseColumnInteger	Retourne la valeur d'une colonne de type int de l'enregistrement courant	Utilisation de bases de données
DatabaseColumnLong	Retourne la valeur d'une colonne de type long de l'enregistrement courant	Utilisation de bases de données
DatabaseColumnDouble	Retourne la valeur d'une colonne de type double de l'enregistrement courant	Utilisation de bases de données
DatabaseColumnBlob	Retourne le tableau de caractères d'une colonne de type BLOB de l'enregistrement courant	Utilisation de bases de données
DebugBreak	Le point d'arrêt de programme de débogage	Fonctions communes
Digits	Rend le nombre de signes décimaux après la virgule, définissant l'exactitude de la mesure du prix du symbole du graphique courant	Vérification de l'état

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
DoubleToString	La conversion de la valeur numérique à la chaîne de texte avec l'exactitude indiquée	Conversion des données
DXContextCreate	Crée un contexte gaphique pour le rendu des cadres de la taille spécifiée	Utilisation de DirectX
DXContextSetSize	Change la taille du cadre du contexte graphique créé avec DXContextCreate()	Utilisation de DirectX
DXContextGetSize	Retourne la taille du cadre d'un contexte graphique créé avec DXContextCreate()	Utilisation de DirectX
DXContextClearColors	Applique la couleur spécifiée à tous les pixels pour le buffer de rendu	Utilisation de DirectX
DXContextClearDepth	Efface le buffer de profondeur	Utilisation de DirectX
DXContextGetColors	Retourne l'image de la taille et du décalage spécifiés pour un contexte graphique	Utilisation de DirectX
DXContextGetDepth	Retourne le buffer de profondeur du cadre rendu	Utilisation de DirectX
DXBufferCreate	Crée le buffer d'un type spécifié basé sur un tableau de données	Utilisation de DirectX
DXTextureCreate	Crée une texture 2D à partir d'un rectangle de la taille spécifiée à partir de l'image passée	Utilisation de DirectX
DXInputCreate	Crée les entrées d'un shader	Utilisation de DirectX
DXInputSet	Définit les entrées d'un shader	Utilisation de DirectX
DXShaderCreate	Crée un shader du type spécifié	Utilisation de DirectX
DXShaderSetLayout	Définit la mise en page du sommet pour le shader du sommet	Utilisation de DirectX
DXShaderInputsSet	Définit les entrées d'un shader	Utilisation de DirectX
DXShaderTexturesSet	Définit les textures du shader	Utilisation de DirectX
DXDraw	Effectue le rendering des sommets du buffer du sommet définit avec DXBufferSet()	Utilisation de DirectX

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
DXDrawIndexed	Effectue le rendering des primitives graphiques décrites par le buffer d'index depuis DXBufferSet()	Utilisation de DirectX
DXPrimiveTopologySet	Définit le type des primitives du rendu en utilisant DXDrawIndexed()	Utilisation de DirectX
DXBufferSet	Définit le buffer pour le rendu courant	Utilisation de DirectX
DXShaderSet	Définit un shader pour le rendu	Utilisation de DirectX
DXHandleType	Retourne le type du handle	Utilisation de DirectX
DXRelease	Libère un handle	Utilisation de DirectX
EnumToString	La conversion de la valeur d'un transfert de n'importe quel type dans une chaîne	Conversion des données
EventChartCustom	Génère l'événement d'utilisateur pour le graphique indiqué	Travail avec les événements
EventKillTimer	Arrête sur le graphique courant la génération des événements selon le minuteur	Travail avec les événements
EventSetMillisecondTimer	Lance le générateur des événements de la minuterie d'une haute permission avec la période moins 1 seconde pour un graphique courant	Travail avec les événements
EventSetTimer	Lance le générateur des événements du minuteur avec la périodicité indiquée pour le graphique courant	Travail avec les événements
exp	Rend l'exposant au nombre	Fonctions mathématiques
ExpertRemove	Arrête le travail de l'expert et le décharge du graphique	Fonctions communes
fabs	Rend la valeur absolue (la valeur selon le module) du nombre transmis à elle	Fonctions mathématiques
FileClose	Ferme le fichier auparavant ouvert	Opérations de fichier
FileCopy	Copie le fichier original d'un répertoire local ou commun à un	Opérations de fichier

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	autre fichier	
FileDelete	Supprime le fichier indiqué	Opérations de fichier
FileFindClose	Ferme le handle de la recherche	Opérations de fichier
FileFindFirst	Commence l'excédent des fichiers dans le répertoire correspondant conformément au filtre indiqué	Opérations de fichier
FileFindNext	Continue la recherche commencée par la fonction FileFindFirst()	Opérations de fichier
FileFlush	Écrit à un disque toutes les données restant dans le tampon du fichier d'entrée-de sortie	Opérations de fichier
FileGetInteger	Reçoit la propriété entière du fichier	Opérations de fichier
FileIsEnding	Définit la fin du fichier en train de la lecture	Opérations de fichier
FileIsExist	Vérifie l'existence d'un fichier	Opérations de fichier
FileIsLineEnding	Définit la fin de la ligne dans le fichier du texte en train de la lecture	Opérations de fichier
FileMove	Déplace et renomme le fichier	Opérations de fichier
FileOpen	Ouvre un fichier avec un nom et un drapeau indiqués	Opérations de fichier
FileReadArray	Lit les tableaux de tous les types, sauf les tableaux de chaîne (peut être un tableau des structures qui ne contient pas les chaînes et les tableaux dynamiques), du fichier binaire de la position courante de l'indicateur de fichier	Opérations de fichier
FileReadBool	Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type bool	Opérations de fichier
FileReadDatetime	Lit du fichier du type CSV la chaîne d'un des formats: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" ou "HH:MI:SS" - et	Opérations de fichier

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	la transforme en valeur du type datetime	
FileReadDouble	Lit le nombre de l'exactitude double avec une virgule flottante (double) du fichier binaire de la position courante de l'indicateur de fichier	Opérations de fichier
FileReadFloat	Lit de la position courante de l'indicateur de fichier la valeur du type float	Opérations de fichier
FileReadInteger	Lit du fichier binaire la valeur du type int, short ou char en fonction de la longueur indiquée dans les bytes	Opérations de fichier
FileReadLong	Lit de la position courante de l'indicateur de fichier la valeur du type long	Opérations de fichier
FileReadNumber	Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type double	Opérations de fichier
FileReadString	Lit du fichier la chaîne de la position courante de l'indicateur de fichier	Opérations de fichier
FileReadStruct	Lit du fichier binaire le contenu à la structure transmise à titre du paramètre	Opérations de fichier
FileSeek	Déplace la position de l'indicateur de fichier par un nombre spécifié de bytes par rapport à la position indiquée	Opérations de fichier
FileSize	Rend la dimension d'un fichier ouvert correspondant	Opérations de fichier
FileTell	Rend la position courante de l'indicateur de fichier du fichier correspondant ouvert	Opérations de fichier
FileWrite	Inscrit les données au fichier comme CSV ou TXT	Opérations de fichier

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
FileWriteArray	Inscrit au fichier du type BIN les tableaux des tous les types, sauf les tableaux de chaîne	Opérations de fichier
FileWriteDouble	Inscrit au fichier binaire la valeur du paramètre du type double de la position courante de l'indicateur de fichier	Opérations de fichier
FileWriteFloat	Inscrit au fichier binaire la valeur du paramètre du type float de la position courante de l'indicateur de fichier	Opérations de fichier
FileWriteInteger	Inscrit au fichier binaire la valeur du paramètre du type int de la position courante de l'indicateur de fichier	Opérations de fichier
FileWriteLong	Inscrit au fichier binaire la valeur du paramètre du type long de la position courante de l'indicateur de fichier	Opérations de fichier
FileWriteString	Inscrit au fichier du type BIN ou TXT la valeur du paramètre du type string de la position courante de l'indicateur de fichier	Opérations de fichier
FileWriteStruct	Inscrit au fichier binaire le contenu de la structure, transmise à titre du paramètre de la position courante de l'indicateur de fichier	Opérations de fichier
floor	Rend la valeur entière numérique la plus proche par dessous	Fonctions mathématiques
fmax	Rend la valeur maximal des deux valeurs numériques	Fonctions mathématiques
fmin	Rend la valeur minimal des deux valeurs numériques	Fonctions mathématiques
fmod	Rend le reste réel après la division de deux nombres	Fonctions mathématiques
FolderClean	Supprime tous les fichiers dans le dossier indiqué	Opérations de fichier
FolderCreate	Crée le répertoire dans le dossier Files (en fonction de la valeur common_flag)	Opérations de fichier

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
FolderDelete	Supprime le répertoire indiqué. Si le dossier n'est pas vide, elle ne peut pas être supprimé	Opérations de fichier
FrameAdd	Ajoute la trame avec les données	Travail avec les résultats de l'optimisation
FrameFilter	Définit le filtre de la lecture des trames et met l'indicateur au début	Travail avec les résultats de l'optimisation
FrameFirst	Convertit l'indicateur de la lecture des trames au début et oblitére un filtre établi.	Travail avec les résultats de l'optimisation
FrameInputs	Reçoit les paramètres input, sur lesquels la trame a été formé	Travail avec les résultats de l'optimisation
FrameNext	Lit la trame et déplace l'indicateur au suivant	Travail avec les résultats de l'optimisation
GetLastError	Rend la valeur de la dernière erreur	Vérification de l'état
GetPointer	Rend l'indicateur de l'objet	Fonctions communes
GetTickCount	Rend la quantité des millisecondes qui ont passé dès le moment du départ du système	Fonctions communes
GlobalVariableCheck	Contrôle l'existence de la variable globale avec le nom indiqué	Variables globales du terminal de client
GlobalVariableDel	Supprime la variable globale	Variables globales du terminal de client
GlobalVariableGet	Demande la valeur de la variable globale	Variables globales du terminal de client
GlobalVariableName	Rend le nom de la variable globale selon le numéro d'ordre dans le répertoire des variables globales	Variables globales du terminal de client
GlobalVariablesDeleteAll	Supprime les variables globales avec le préfixe indiqué dans le nom	Variables globales du terminal de client
GlobalVariableSet	Met la nouvelle valeur à une variable globale	Variables globales du terminal de client
GlobalVariableSetOnCondition	Met une nouvelle valeur de la variable existant globale selon la condition	Variables globales du terminal de client

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
GlobalVariablesFlush	Inscrit forcément le contenu de toutes les variables globales sur le disque	Variables globales du terminal de client
GlobalVariablesTotal	Rend le total des variables globales	Variables globales du terminal de client
GlobalVariableTemp	Met une nouvelle valeur de la variable globale, qui existe seulement pendant la session courante du terminal	Variables globales du terminal de client
GlobalVariableTime	Rend le temps du dernier accès à la variable globale	Variables globales du terminal de client
HistoryDealGetDouble	Rend la propriété demandée du marché dans l'histoire (double)	Fonctions commerciales
HistoryDealGetInteger	Rend la propriété demandée du marché dans l'histoire (datetime ou int)	Fonctions commerciales
HistoryDealGetString	Rend la propriété demandée du marché dans l'histoire (string)	Fonctions commerciales
HistoryDealGetTicket	Choisit le marché pour le traitement ultérieur et rend le ticket les marchés dans l'histoire	Fonctions commerciales
HistoryDealSelect	Choisit le marché dans l'histoire pour davantage l'appeler par les fonctions appropriées	Fonctions commerciales
HistoryDealsTotal	Rend le nombre de marchés dans l'histoire	Fonctions commerciales
HistoryOrderGetDouble	Rend la propriété demandée de l'ordre dans l'histoire (double)	Fonctions commerciales
HistoryOrderGetInteger	Rend la propriété demandée de l'ordre dans l'histoire (datetime ou int)	Fonctions commerciales
HistoryOrderGetString	Rend la propriété demandée de l'ordre dans l'histoire (string)	Fonctions commerciales
HistoryOrderGetTicket	Rend le ticket de l'ordre correspondant dans l'histoire	Fonctions commerciales
HistoryOrderSelect	Choisit à l'histoire l'ordre pour le travail ultérieur avec lui	Fonctions commerciales
HistoryOrdersTotal	Rend le nombre d'ordres dans l'histoire	Fonctions commerciales

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
HistorySelect	Demande l'histoire des marchés et des ordres pour la période indiquée du temps de serveur	Fonctions commerciales
iBars	Retourne le nombre de barres dans l'historique pour le symbole et la période spécifiés	Accès aux Séries de Données et aux Indicateurs
iBarShift	Retourne l'indice de la barre correspondant à l'heure spécifiée	Accès aux Séries de Données et aux Indicateurs
iClose	Retourne le prix de clôture (Close) de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant	Accès aux Séries de Données et aux Indicateurs
iHigh	Retourne le prix le plus haut (High) de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant	Accès aux Séries de Données et aux Indicateurs
iHighest	Retourne l'indice de la valeur la plus haute trouvée sur le graphique correspondant (décalage relatif à la barre courante)	Accès aux Séries de Données et aux Indicateurs
iLow	Retourne le prix le plus bas (Low) de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant	Accès aux Séries de Données et aux Indicateurs
iLowest	Retourne l'indice de la valeur la plus basse trouvée sur le graphique correspondant (décalage relatif à la barre courante)	Accès aux Séries de Données et aux Indicateurs
iOpen	Retourne le prix d'ouverture (Open) de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant	Accès aux Séries de Données et aux Indicateurs
iTime	Retourne l'heure d'ouverture de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant	Accès aux Séries de Données et aux Indicateurs
iTickVolume	Retourne le volume de ticks de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant	Accès aux Séries de Données et aux Indicateurs

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
iRealVolume	Retourne le volume réel de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant	Accès aux Séries de Données et aux Indicateurs
iVolume	Retourne le volume de ticks de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant	Accès aux Séries de Données et aux Indicateurs
iSpread	Retourne la valeur du spread de la barre (indiquée par le paramètre 'shift') sur le graphique correspondant	Accès aux Séries de Données et aux Indicateurs
iAC	Accelerator Oscillator	Indicateurs techniques
iAD	Accumulation/Distribution	Indicateurs techniques
iADX	Average Directional Index	Indicateurs techniques
iADXWilder	Average Directional Index by Welles Wilder	Indicateurs techniques
iAlligator	Alligator	Indicateurs techniques
iAMA	Adaptive Moving Average	Indicateurs techniques
iAO	Awesome Oscillator	Indicateurs techniques
iATR	Average True Range	Indicateurs techniques
iBands	Bollinger Bands®	Indicateurs techniques
iBearsPower	Bears Power	Indicateurs techniques
iBullsPower	Bulls Power	Indicateurs techniques
iBWMFI	Market Facilitation Index by Bill Williams	Indicateurs techniques
iCCI	Commodity Channel Index	Indicateurs techniques
iChaikin	Chaikin Oscillator	Indicateurs techniques
iCustom	l'indicateur d'utilisateur	Indicateurs techniques
iDEMA	Double Exponential Moving Average	Indicateurs techniques
iDeMarker	DeMarker	Indicateurs techniques
iEnvelopes	Envelopes	Indicateurs techniques
iForce	Force Index	Indicateurs techniques
iFractals	Fractals	Indicateurs techniques

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
iFrAMA	Fractal Adaptive Moving Average	Indicateurs techniques
iGatore	Gator Oscillator	Indicateurs techniques
iIchimoku	Ichimoku Kinko Hyo	Indicateurs techniques
iMA	Moving Average	Indicateurs techniques
iMACD	Moving Averages Convergence-Divergence	Indicateurs techniques
iMFI	Money Flow Index	Indicateurs techniques
iMomentum	Momentum	Indicateurs techniques
IndicatorCreate	Rend le handle de l'indicateur indiqué technique créé à la base du tableau des paramètres du type MqlParam	Accès aux timeséries et les données des indicateurs
IndicatorParameters	Rend le nombre de paramètres d'entrée de l'indicateur selon le handle indiqué, ainsi que les valeurs elles-mêmes et le type des paramètres	Accès aux timeséries et les données des indicateurs
IndicatorRelease	Supprime le handle de l'indicateur et libère la partie calculée de l'indicateur, si personne ne se sert plus d'elle	Accès aux timeséries et les données des indicateurs
IndicatorSetDouble	Spécifie la valeur de la propriété d'indicateur, ayant le type double	Indicateurs d'utilisateur
IndicatorSetInteger	Spécifie la valeur de la propriété d'indicateur, ayant le type int	Indicateurs d'utilisateur
IndicatorSetString	Spécifie la valeur de la propriété d'indicateur, ayant le type string	Indicateurs d'utilisateur
IntegerToString	La conversion de la valeur du type entier à la chaîne de la longueur indiquée	Conversion des données
iOBV	On Balance Volume	Indicateurs techniques
iOsMA	Moving Average of Oscillator (MACD histogram)	Indicateurs techniques
iRSI	Relative Strength Index	Indicateurs techniques
iRVI	Relative Vigor Index	Indicateurs techniques
iSAR	Parabolic Stop And Reverse System	Indicateurs techniques

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
IsStopped	Rend true, si on a donné l'ordre de terminer l'exécution du programme mql5	Vérification de l'état
iStdDev	Standard Deviation	Indicateurs techniques
iStochastic	Stochastic Oscillator	Indicateurs techniques
iTEMA	Triple Exponential Moving Average	Indicateurs techniques
iTriX	Triple Exponential Moving Averages Oscillator	Indicateurs techniques
iVIDyA	Variable Index Dynamic Average	Indicateurs techniques
iVolumes	Volumes	Indicateurs techniques
iWPR	Williams' Percent Range	Indicateurs techniques
log	Rend le logarithme naturel	Fonctions mathématiques
log10	Rend le logarithme du nombre selon le fondement 10	Fonctions mathématiques
MarketBookAdd	Assure l'ouverture du profondeur de marché selon l'instrument indiqué, ainsi que produit la souscription sur la réception des avis du changement du profondeur de marché indiqué	Réception de l'information de marché
MarketBookGet	Rend le tableau des structures du type MqlBookInfo, contenant les enregistrements du profondeur de marché du symbole indiqué	Réception de l'information de marché
MarketBookRelease	Assure la clôture du profondeur de marché selon l'instrument, ainsi qui élimine la souscription sur la réception des avis du changement du profondeur de marché indiqué	Réception de l'information de marché
MathAbs	Rend la valeur absolue (la valeur selon le module) du nombre transmis à elle	Fonctions mathématiques
MathArccos	Rend la valeur d'arc cosinus x aux radians	Fonctions mathématiques
MathArcsin	Rend la valeur d'arc sinus x aux radians	Fonctions mathématiques
MathArctan	Rend l'arc tangent x aux radians	Fonctions mathématiques

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
MathCeil	Rend la valeur entière numérique la plus proche par dessus	Fonctions mathématiques
MathCos	Rend le cosinus du nombre	Fonctions mathématiques
MathExp	Rend l'exposant au nombre	Fonctions mathématiques
MathFloor	Rend la valeur entière numérique la plus proche par dessous	Fonctions mathématiques
MathIsValidNumber	Vérifie l'exactitude d'un nombre réel	Fonctions mathématiques
MathLog	Rend le logarithme naturel	Fonctions mathématiques
MathLog10	Rend le logarithme du nombre selon le fondement 10	Fonctions mathématiques
MathMax	Rend la valeur maximal des deux valeurs numériques	Fonctions mathématiques
MathMin	Rend la valeur minimal des deux valeurs numériques	Fonctions mathématiques
MathMod	Rend le reste réel après la division de deux nombres	Fonctions mathématiques
MathPow	Elève le fondement à la puissance indiquée	Fonctions mathématiques
MathRand	Rend le nombre entier pseudo-accidentelle dans le domaine de 0 jusqu'à 32767	Fonctions mathématiques
MathRound	Arrondit le nombre jusqu'à l'entier plus proche	Fonctions mathématiques
MathSin	Rend le sinus du nombre	Fonctions mathématiques
MathSqrt	Rend la racine carrée	Fonctions mathématiques
MathSrand	Établit l'état initial du générateur des nombres entiers quasi-aléatoires	Fonctions mathématiques
MathTan	Rend la tangent du nombre	Fonctions mathématiques
MessageBox	Crée, affiche la fenêtre des messages et la dirige	Fonctions communes
MQLInfoInteger	Rend la valeur du type entier de la propriété correspondante du programme mql5 lancé	Vérification de l'état
MQLInfoString	Rend la valeur du type string de la propriété correspondante du	Vérification de l'état

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	programme mql5 lancé	
MT5Initialize	Etablit une connexion avec le terminal MetaTrader 5	MetaTrader pour Python
MT5Shutdown	Ferme la connexion avec le terminal MetaTrader 5	MetaTrader pour Python
MT5TerminalInfo	Retourne le statut et les paramètres du terminal MetaTrader 5 connecté	MetaTrader pour Python
MT5Version	Retourne la version du terminal MetaTrader 5	MetaTrader pour Python
MT5CopyRatesFrom	Retourne les barres du terminal MetaTrader 5 commençant à partir de la date spécifiée	MetaTrader pour Python
MT5CopyRatesFromPos	Retourne les barres du terminal MetaTrader 5 commençant à partir de l'indice spécifié	MetaTrader pour Python
MT5CopyRatesRange	Retourne les barres du terminal MetaTrader 5 dans l'intervalle de dates spécifiées	MetaTrader pour Python
MT5CopyTicksFrom	Retourne les ticks du terminal MetaTrader 5 commençant à la date spécifiée	MetaTrader pour Python
MT5CopyTicksRange	Retourne les ticks du terminal MetaTrader 5 pour l'intervalle de dates spécifiées	MetaTrader pour Python
NormalizeDouble	L'arrondissement du nombre de la virgule flottante à l'exactitude indiquée	Conversion des données
ObjectCreate	Crée l'objet du type spécifié sur le graphique indiqué	Objets graphiques
ObjectDelete	Supprime l'objet avec le nom indiqué du graphique indiqué (de la sous-fenêtre de graphique indiquée)	Objets graphiques
ObjectFind	Cherche l'objet selon le nom avec l'identificateur indiqué	Objets graphiques
ObjectGetDouble	Rend la valeur du type double de la propriété correspondante de l'objet	Objets graphiques

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
ObjectGetInteger	Rend la valeur entière de la propriété correspondante de l'objet	Objets graphiques
ObjectGetString	Rend la valeur du type string de la propriété correspondante de l'objet	Objets graphiques
ObjectGetTimeByValue	Rend la valeur du temps pour la valeur indiquée du prix de l'objet	Objets graphiques
ObjectGetValueByTime	Rend la valeur de prix de l'objet pour le temps indiqué	Objets graphiques
ObjectMove	Change les coordonnées du point indiqué du rattachement de l'objet	Objets graphiques
ObjectName	Rend le nom de l'objet du type correspondant dans le graphique indiqué (indiqué dans la sous-fenêtre du graphique)	Objets graphiques
ObjectsDeleteAll	Supprime tous les objets du type indiqué du graphique indiqué (de la sous-fenêtre de graphique indiquée)	Objets graphiques
ObjectSetDouble	Met la valeur de la propriété correspondante de l'objet	Objets graphiques
ObjectSetInteger	Met la valeur de la propriété correspondante de l'objet	Objets graphiques
ObjectSetString	Met la valeur de la propriété correspondante de l'objet	Objets graphiques
ObjectsTotal	Rëend le nombre d'objets du type indiqué dans le graphique indiqué (la sous-fenêtre de graphique indiquée)	Objets graphiques
OnStart	La fonction est appelée lorsque l'évènement Start se produit pour effectuer les actions définies dans le script	Gestion d'Evènements
OnInit	La fonction est appelée dans les indicateurs et les EA lorsque l'évènement Init se produit pour initialiser un programme MQL5 lancé	Gestion d'Evènements
OnDeinit	La fonction est appelée dans les indicateurs et les EA lorsque	Gestion d'Evènements

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	l'évèenement Deinit se produit pour désinitialiser un programme MQL5 lancé	
OnTick	La fonction est appelée dans les EA lorsque l'évènement NewTick se produit pour gérer une nouvelle cotation	Gestion d'Evènements
OnCalculate	La fonction est appelée dans les indicateurs lorsque l'évènement Calculate se produit pour gérer les changements de prix	Gestion d'Evènements
OnTimer	La fonction est appelée dans les indicateurs et les EA pendant l'évènement périodique Timer généré par le terminal à des intervalles fixes de temps	Gestion d'Evènements
OnTrade	La fonction est appelée dans les EA pendant l'évènement Trade généré à la fin d'une opération de trading sur un serveur de trades	Gestion d'Evènements
OnTradeTransaction	La fonction est appelée dans les EA lorsque l'évènement TradeTransaction se produit pour traiter les résultats de l'exécution d'une demande de trade	Gestion d'Evènements
OnBookEvent	La fonction est appelée dans les EA lorsque l'évènement BookEvent se produit pour traiter les changements du market depth	Gestion d'Evènements
OnChartEvent	La fonction est appelée dans les indicateurs et les EA lorsque l'évènement ChartEvent se produit pour traiter les changements du graphique effectués par un utilisateur ou un programme MQL5	Gestion d'Evènements
OnTester	La fonction est appelée dans les EA lorsque l'évènement Tester se produit pour effectuer les actions nécessaires après le test d'un EA sur les données historiques	Gestion d'Evènements
OnTesterInit	La fonction est appelée dans les EA lorsque l'évènement TesterInit	Gestion d'Evènements

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	se produit pour effectuer les actions nécessaires avant une optimisation dans le testeur de stratégies	
OnTesterDeinit	La fonction est appelée dans les EA lorsque l'évènement TesterDeinit se produit après l'optimisation d'un EA dans le testeur de stratégie	Gestion d'Evènements
OnTesterPass	La fonction est appelée dans les EA lorsque l'évènement TesterPass se produit pour gérer l'arrivée d'un nouvel ensemble de données pendant l'optimisation d'un EA dans le testeur de stratégie	Gestion d'Evènements
OrderCalcMargin	Calcule le montant de la marge nécessaire au type indiqué de l'ordre, en devise du compte	Fonctions commerciales
OrderCalcProfit	Calcule le montant du bénéfice en vertu des paramètres transmis en devise du compte	Fonctions commerciales
OrderCheck	Contrôle la suffisance des moyens pour l'accomplissement de la transaction commerciale demandée.	Fonctions commerciales
OrderGetDouble	Rend la propriété demandée de l'ordre (double)	Fonctions commerciales
OrderGetInteger	Rend la propriété demandée de l'ordre (datetime ou int)	Fonctions commerciales
OrderGetString	Rend la propriété demandée de l'ordre (string)	Fonctions commerciales
OrderGetTicket	Rend le ticket de l'ordre correspondant	Fonctions commerciales
OrderSelect	Choisit l'ordre pour le travail ultérieur avec lui	Fonctions commerciales
OrderSend	Expédie les demandes commerciales au serveur	Fonctions commerciales
OrderSendAsync	Envoie les demandes commerciales asynchronement	Fonctions commerciales

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	sans attente de la réponse du serveur commercial	
OrdersTotal	Rend le nombre d'ordres	Fonctions commerciales
ParameterGetRange	Reçoit pour la variable inputl'information sur la gamme des valeurs et sur le pas du changement à l'optimisation de l'expert dans le testeur des stratégies	Travail avec les résultats de l'optimisation
ParameterSetRange	Établit des règles de l'utilisation de la variable inputà l'optimisation de l'expert dans le testeur des stratégies: la valeur, le pas du changement, les valeurs initiale et finale	Travail avec les résultats de l'optimisation
Period	Rend la valeur du temps trame du graphique courant	Vérification de l'état
PeriodSeconds	Rend la quantité de secondes dans la période	Fonctions communes
PlaySound	Reproduit le fichier sonore	Fonctions communes
PlotIndexGetInteger	Rend la valeur de la propriété de la ligne d'indicateur, ayant le type entier	Indicateurs d'utilisateur
PlotIndexSetDouble	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type double	Indicateurs d'utilisateur
PlotIndexSetInteger	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type int	Indicateurs d'utilisateur
PlotIndexSetString	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type string	Indicateurs d'utilisateur
Point	Rend la dimension de point de l'instrument courant en devise de la cotation	Vérification de l'état
PositionGetDouble	Rend la propriété demandée de la position ouverte (double)	Fonctions commerciales
PositionGetInteger	Rend la propriété demandée de la position ouverte (datetime ou int)	Fonctions commerciales

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
PositionGetString	Rend la propriété demandée de la position ouverte (string)	Fonctions commerciales
PositionGetSymbol	Rend le symbole de la position correspondante ouverte	Fonctions commerciales
PositionSelect	Choisit la position ouverte pour le travail ultérieur avec elle	Fonctions commerciales
PositionsTotal	Rend le nombre de positions ouvertes	Fonctions commerciales
pow	Elève le fondement à la puissance indiquée	Fonctions mathématiques
Print	Affiche un message dans le journal	Fonctions communes
PrintFormat	Formate et imprime l'ensemble des symboles et des valeurs dans un fichier-journal conformément au format donné	Fonctions communes
rand	Rend le nombre entier pseudo-accidentelle dans le domaine de 0 jusqu'à 32767	Fonctions mathématiques
ResetLastError	Établit la valeur de la variable prédéterminée _LastError au zéro	Fonctions communes
ResourceCreate	Crée la ressource de l'image à la base de l'ensemble des données	Fonctions communes
ResourceFree	Supprime la ressource dynamiquement créée (libère la mémoire occupée par la ressource)	Fonctions communes
ResourceReadImage	Lit les données de la ressource graphique, créé par la fonction ResourceCreate() ou sauvegardée dans le fichier EX5 à la compilation	Fonctions communes
ResourceSave	Sauvegarde la ressource au fichier spécifié	Fonctions communes
round	Arrondit le nombre jusqu'à l'entier plus proche	Fonctions mathématiques
SendFTP	Envoie le fichier à l'adresse indiquée dans la fenêtre des réglages du signet "la publication"	Fonctions communes

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
SendMail	Envoie le message électronique à l'adresse indiquée dans la fenêtre des réglages du signet "la Poste"	Fonctions communes
SendNotification	Envoie les notifications Push aux terminaux mobiles, dont les ID MetaQuotes sont indiqués sur l'onglet "Notifications"	Fonctions communes
SeriesInfoInteger	Rend l'information sur l'état des données historiques	Accès aux timeséries et les données des indicateurs
SetIndexBuffer	Attache le tampon d'indicateur indiquée avec le tableau unidimentionnel dynamique du type double	Indicateurs d'utilisateur
ShortArrayToString	Copie la partie du tableau dans la chaîne	Conversion des données
ShortToString	La conversion du code de symbole (unicode)à la chaîne d'un symbole	Conversion des données
SignalBaseGetDouble	Возвращает значение свойства типа double для выбранного сигнала	Trade Signals
SignalBaseGetInteger	Возвращает значение свойства типа integer для выбранного сигнала	Trade Signals
SignalBaseGetString	Возвращает значение свойства типа string для выбранного сигнала	Trade Signals
SignalBaseSelect	Выбирает для работы сигнал из базы торговых сигналов, доступных в терминале	Trade Signals
SignalBaseTotal	Возвращает общее количество сигналов, доступных в терминале	Trade Signals
SignalInfoGetDouble	Возвращает из настроек копирования торгового сигнала значение свойства типа double	Trade Signals
SignalInfoGetInteger	Возвращает из настроек копирования торгового сигнала значение свойства типа integer	Trade Signals
SignalInfoGetString	Возвращает из настроек копирования торгового сигнала значение свойства типа string	Trade Signals

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
SignalInfoSetDouble	Устанавливает в настройках копирования торгового сигнала значение свойства типа double	Trade Signals
SignalInfoSetInteger	Устанавливает в настройках копирования торгового сигнала значение свойства типа integer	Trade Signals
SignalSubscribe	Производит подписку на копирование торгового сигнала	Trade Signals
SignalUnsubscribe	Отменяет подписку на копирование торгового сигнала	Trade Signals
sin	Rend le sinus du nombre	Fonctions mathématiques
Sleep	Retient l'exécution de l'expert courant ou le script sur l'intervalle défini	Fonctions communes
SocketCreate	Crée une socket avec les flags spécifiés et retourne son handle	Fonctions réseau
SocketClose	Ferme une socket	Fonctions réseau
SocketConnect	Connexion au serveur avec un contrôle du timeout	Fonctions réseau
SocketIsConnected	Vérifie si la socket est actuellement connectée	Fonctions réseau
SocketIsReadable	Retourne le nombre d'octets pouvant être lus sur une socket	Fonctions réseau
SocketIsWritable	Vérifie si des données peuvent être écrites actuellement sur une socket	Fonctions réseau
SocketTimeouts	Définit les timeouts pour recevoir et envoyer des données sur une socket système	Fonctions réseau
SocketRead	Lit les données d'une socket	Fonctions réseau
SocketSend	Ecrit des données sur une socket	Fonctions réseau
SocketTlsHandshake	Initie une connexion TLS (SSL) vers un hôte spécifié via le protocole TLS Handshake	Fonctions réseau
SocketTlsCertificate	Retourne les données sur le certificat utilisé pour une connexion sécurisée au réseau	Fonctions réseau

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
SocketTlsRead	Lit des données depuis une connexion TLS sécurisée	Fonctions réseau
SocketTlsReadAvailable	Lit toutes les données disponibles depuis une connexion TLS sécurisée	Fonctions réseau
SocketTlsSend	Envoi des données via une connexion TLS sécurisée	Fonctions réseau
sqrt	Rend la racine carrée	Fonctions mathématiques
srand	Établit l'état initial du générateur des nombres entiers quasi-aléatoires	Fonctions mathématiques
StringAdd	Ajoute la chaîne selon la place indiquée par la sous-chaîne	Fonctions de chaîne
StringBufferLen	Rend la grandeur du buffer distribué pour la chaîne.	Fonctions de chaîne
StringCompare	Compare deux chaînes et rend 1 si la première chaîne est plus que deuxième; 0 - si les chaînes sont égales; -1 (moins un) - si la première chaîne est moins que la deuxième	Fonctions de chaîne
StringConcatenate	Forme la chaîne des paramètres transmis	Fonctions de chaîne
StringFill	Remplit la chaîne indiquée selon la place par les symboles indiqués	Fonctions de chaîne
StringFind	La recherche de la sous-chaîne dans la chaîne	Fonctions de chaîne
StringFormat	Convertit le nombre à la chaîne conformément au format spécifié	Conversion des données
StringGetCharacter	Rend la valeur du symbole disposé dans la position indiquée de la chaîne	Fonctions de chaîne
StringInit	Initialise la chaîne par les symboles indiqués et assure la grandeur indiquée de la chaîne	Fonctions de chaîne
StringLen	Rend le nombre de symboles dans la chaîne	Fonctions de chaîne
StringReplace	Remplace tous les sous-chaînes recherchés dans la chaîne pour la	Fonctions de chaîne

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	séquence spécifiée des caractères.	
StringSetCharacter	Rend la copie de la chaîne avec la valeur changée du symbole dans la position indiquée	Fonctions de chaîne
StringSplit	Reçoit les sous-chaînes de la chaîne indiquée par un séparateur spécifié et rend le montant des sous-chaînes reçus	Fonctions de chaîne
StringSubstr	Extrait la sous-chaîne de la chaîne de texte qui commence de la position indiquée	Fonctions de chaîne
StringToCharArray	Copie par un symbole la chaîne convertie de Unicode à ANSI à l'endroit spécifié du tableau du type uchar	Conversion des données
StringToColor	Convertit la chaîne du type "R,G,B" ou la chaîne, contenant le nom de la couleur, à la valeur du type color	Conversion des données
StringToDouble	La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type double	Conversion des données
StringToInteger	La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type int	Conversion des données
StringToLower	Transforme tous les symboles de la chaîne indiquée aux symboles minuscules selon la place	Fonctions de chaîne
StringToShortArray	Copie la chaîne par un symbole à l'endroit spécifié du tableau du type ushort	Conversion des données
StringToTime	La conversion de la chaîne, contenant le temps et/ou la date dans le format "yyyy.mm.dd [hh:mi]", dans le nombre du type datetime	Conversion des données
StringToUpper	Transforme tous les symboles de la chaîne indiquée aux symboles majuscules selon la place	Fonctions de chaîne

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
StringTrimLeft	Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation du début de la chaîne	Fonctions de chaîne
StringTrimRight	Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation à la fin de la chaîne	Fonctions de chaîne
StructToTime	Produit la conversion de la valeur du type de la structure MqlDateTime à la variable du type datetime	Date et temps
Symbol	Rend le nom du symbole du graphique courant	Vérification de l'état
SymbolInfoDouble	Rend la valeur du type double du symbole indiqué pour la propriété correspondante	Réception de l'information de marché
SymbolInfoInteger	Rend la valeur du type entier (long, datetime, int ou bool) du symbole indiqué pour la propriété correspondante	Réception de l'information de marché
SymbolInfoMarginRate	Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера	Réception de l'information de marché
SymbolInfoSessionQuote	Permet de recevoir le temps du début et le temps de la fin de la session de cotation indiquée pour le symbole et le jour de la semaine indiqués.	Réception de l'information de marché
SymbolInfoSessionTrade	Permet de recevoir le temps du début et le temps de la fin de la session commerciale indiquée pour le symbole et le jour de la semaine indiqués.	Réception de l'information de marché
SymbolInfoString	Rend la valeur du type string du symbole indiqué pour la propriété correspondante	Réception de l'information de marché
SymbolInfoTick	Rend les valeurs courantes pour le symbole indiqué dans la variable du type MqlTick	Réception de l'information de marché
SymbolIsSynchronized	Vérifie le synchronismedes données selon le symbole indiqué	Réception de l'information de marché

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	dans le terminal avec les données sur le serveur commercial	
SymbolName	Rend le nom du symbole indiqué	Réception de l'information de marché
SymbolSelect	Choisit le symbole dans la fenêtre MarketWatch ou enlève le symbole de la fenêtre	Réception de l'information de marché
SymbolsTotal	Rend le nombre accessible (choisi dans MarketWatch ou tous)des symboles	Réception de l'information de marché
tan	Rend la tangent du nombre	Fonctions mathématiques
TerminalClose	Envoie au terminal l'ordre sur l'achèvement du travail	Fonctions communes
TerminalInfoDouble	Rend la valeur du type double de la propriété correspondante de l'environnement du programme mql5	Vérification de l'état
TerminalInfoInteger	Rend la valeur du type entier de la propriété correspondante de l'environnement du programme mql5	Vérification de l'état
TerminalInfoString	Rend la valeur du type string de la propriété correspondante de l'environnement du programme mql5	Vérification de l'état
TesterStatistics	Rend la valeur du paramètre indiqué statistique, calculée d'après les résultats du test	Fonctions communes
TextGetSize	Rend la largeur et la hauteur de la ligne aux réglages courants de la fonte	Objets graphiques
TextOut	Déduit le texte au tableau d'utilisateur (le tampon) destiné à la création de la ressourcegraphique	Objets graphiques
TextSetFont	Établit la fonte pour la sortie du texte par les méthodes du dessin (on utilise par défaut la fonte Arial 20)	Objets graphiques
TimeCurrent	Rend le dernier temps connu du serveur (le temps de l'arrivée de	Date et temps

© 2000-2024, MetaQuotes Ltd.

Fonction	Action	Раздел
	la dernière cotation) dans le format datetime	
TimeDaylightSavings	Rend le signe du passage sur l'heure d'été/l'hiver	Date et temps
TimeGMT	Rend le temps GMT au format datetime en tenant compte du passage sur l'heure d'hiver ou d'été par le temps local de l'ordinateur, sur lequel le terminal de client a été lancé	Date et temps
TimeGMTOffset	Rend la différence actuelle entre le temps GMT et le temps local de l'ordinateur en secondes en tenant compte du passage sur l'heure d'hiver ou d'été	Date et temps
TimeLocal	Rend le temps local informatique dans le format datetime	Date et temps
TimeToString	La conversion de la valeur, contenant le temps en secondes, passé après le 01.01.1970, à la chaîne du format "yyyy.mm.dd hh:mi"	Conversion des données
TimeToStruct	Produit la conversion de la valeur du type datetime à la variable du type de la structure MqlDateTime	Date et temps
TimeTradeServer	Rend le temps calculé courant du serveur commercial	Date et temps
UninitializeReason	Rend le code de la raison de la déinitialisation	Vérification de l'état
WebRequest	Отправляет HTTP-запрос на указанный сервер	Fonctions communes
ZeroMemory	Remet au zéro la variable transmise selon la référence. La variable peut être de n'importe quel type, seulement les classes et les structures qui ont les constructeurs font l'exception	Fonctions communes

© 2000-2024, MetaQuotes Ltd.

Fonctions réseau

List of MQL5 Constants

All MQL5 constants in alphabetical order.

Constante	Description	Использование
__DATE__	La date de la compilation du fichier sans l'heure (les heures, les minutes et les secondes sont égales 0)	Print
__DATETIME__	La date et le temps de la compilation du fichier	Print
__FILE__	Le nom du fichier courant compilé	Print
__FUNCSIG__	La signature de la fonction, dont dans le corps la macro est située. La sortie de la description	Print

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	complète de la fonction avec les types des paramètres dans le log peut être utile à l'identification des fonctions surchargées	
__FUNCTION__	Le nom de la fonction, dans le corps duquel se trouve la macro	Print
__LINE__	Le numéro de la ligne dans le code source, où se trouve la macro donnée	Print
__MQLBUILD__, __MQL5BUILD__	Le numéro du build du compilateur	Print

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
__PATH__	Le chemin absolu vers le fichier compilé actuel	Print
ACCOUNT_ASSETS	Текущий размер активов на счёте	AccountInfoDouble
ACCOUNT_BALANCE	La balance du compte en devise du dépôt	AccountInfoDouble
ACCOUNT_COMMISSION_BLOCKED	Текущая сумма заблокированных комиссий по счёту	AccountInfoDouble
ACCOUNT_COMPANY	Le nom d'une compagnie qui sert le compte	AccountInfoString
ACCOUNT_CREDIT	Le montant du crédit accordé en devise du dépôt	AccountInfoDouble
ACCOUNT_CURRENCY	La devise du dépôt	AccountInfoString

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ACCOUNT_EQUITY	La valeur des moyens propres sur le compte en devise du dépôt	AccountInfoDouble
ACCOUNT_LEVERAGE	Le montant de l'épaule accordée	AccountInfoInteger
ACCOUNT_LIABILITIES	Текущий размер обязательств на счёте	AccountInfoDouble
ACCOUNT_LIMIT_ORDERS	La quantité au maximum admissible d'ordres actifs remis	AccountInfoInteger
ACCOUNT_LOGIN	Le numéro de compte	AccountInfoInteger
ACCOUNT_MARGIN	Le montant des moyens réservés hypothécaires sur le compte en	AccountInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	devise du dépôt	
ACCOUNT_MARGIN_FREE	Le montant des moyens libres sur le compte en devise du dépôt, accessible pour l'ouverture de la position	AccountInfoDouble
ACCOUNT_MARGIN_INITIAL	Размер средств, зарезервированных на счёте, для обеспечения гарантийной суммы по всем отложенным ордерам	AccountInfoDouble
ACCOUNT_MARGIN_LEVEL	Le niveau des moyens hypothécaires sur le compte en pourcentage	AccountInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ACCOUNT_MARGIN_MAINTENANCE	Размер средств, зарезервированных на счёте, для обеспечения минимальной суммы по всем открытым позициям	AccountInfoDouble
ACCOUNT_MARGIN_SO_CALL	Margin call level. Depending on the set ACCOUNT_MARGIN_SO_MODE is expressed in percents or in the deposit currency. ACCOUNT_MARGIN_SO_MODE s'exprime en pourcentage ou en devise du dépôt	AccountInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ACCOUNT_MARGIN_SO_MODE	Le mode de la spécification du niveau minimum admissible des moyens hypothécaires	AccountInfoInteger
ACCOUNT_MARGIN_SO_SO	Margin stop out level. Depending on the set ACCOUNT_MARGIN_SO_MODE is expressed in percents or in the deposit currency. ACCOUNT_MARGIN_SO_MODE s'exprime en pourcentage ou en devise du dépôt	AccountInfoDouble
ACCOUNT_NAME	Le nom du client	AccountInfoString
ACCOUNT_PROFIT	Le montant du bénéfice	AccountInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	courant sur le compte en devise du dépôt	
ACCOUNT_SERVER	Le nom du serveur commercial	AccountInfoString
ACCOUNT_STOPOUT_MODE_MONEY	Le niveau est donné dans l'argent	AccountInfoInteger
ACCOUNT_STOPOUT_MODE_PERCENT	Le niveau est donné en pourcentage	AccountInfoInteger
ACCOUNT_TRADE_ALLOWED	Le commerce permis pour le compte courant	AccountInfoInteger
ACCOUNT_TRADE_EXPERT	Le commerce permis pour l'expert	AccountInfoInteger
ACCOUNT_TRADE_MODE	Account trade mode	AccountInfoInteger
ACCOUNT_TRADE_MODE_CONTEST	Le compte de	AccountInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	concours commercial	
ACCOUNT_TRADE_MODE_DEMO	Le compte commercial de démonstration	AccountInfoInteger
ACCOUNT_TRADE_MODE_REAL	Le compte réel commercial	AccountInfoInteger
ALIGN_CENTER	L'alignement selon le centre (seulement pourl'objet "Champ de l'entrée")	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_LEFT	L'alignement selon une frontière gauche	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_RIGHT	L'alignement selon une frontière droite	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ANCHOR_CENTER	Le point du rattachement strictement	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	selon le centre de l'objet	
ANCHOR_LEFT	Le point du rattachement à gauche selon le centre	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_LOWER	Le point du rattachement dans un angle gauche inférieur	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_UPPER	Le point du rattachement dans un angle gauche supérieur	ObjectSetInteger, ObjectGetInteger
ANCHOR_LOWER	Le point du rattachement d'en bas selon le centre	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT	Le point du rattachement à droite selon le centre	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_LOWER	Le point du	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	rattachement dans un angle droit inférieur	
ANCHOR_RIGHT_UPPER	Le point du rattachement à droite selon le centre	ObjectSetInteger, ObjectGetInteger
ANCHOR_UPPER	Le point du rattachement par dessus selon le centre	ObjectSetInteger, ObjectGetInteger
BASE_LINE	Une ligne principale	Lignes des indicateurs
BOOK_TYPE_BUY	La demande pour l'achat	MqlBookInfo
BOOK_TYPE_BUY_MARKET	La demande pour l'achat au prix du marché	MqlBookInfo
BOOK_TYPE_SELL	La demande pour la vente	MqlBookInfo
BOOK_TYPE_SELL_MARKET	La demande pour la	MqlBookInfo

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	vente au prix du marché	
BORDER_FLAT	Forme plate	ObjectSetInteger, ObjectGetInteger
BORDER_RAISED	Forme proéminente	ObjectSetInteger, ObjectGetInteger
BORDER_SUNKEN	Forme concave	ObjectSetInteger, ObjectGetInteger
CHAR_MAX	La valeur maximale, qui peut être présentée par le type char	Constantes des types de données numériques
CHAR_MIN	La valeur minimale, qui peut être présentée par le type char	Constantes des types de données numériques
CHART_AUTOSCROLL	Le mode du déplacement automatique vers le bord droit du graphique	ChartSetInteger, ChartGetInteger
CHART_BARS	L'affichage	ChartSetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	comme les barres	
CHART_BEGIN	Le début du graphique (les prix les plus vieux)	ChartNavigate
CHART_BRING_TO_TOP	L'affichage du graphique au-dessus de tous les autres	ChartSetInteger, ChartGetInteger
CHART_CANDLES	L'affichage comme les chandeliers japonais	ChartSetInteger
CHART_COLOR_ASK	La couleur de la ligne du prix Ask	ChartSetInteger, ChartGetInteger
CHART_COLOR_BACKGROUND	La couleur du fond du graphique	ChartSetInteger, ChartGetInteger
CHART_COLOR_BID	La couleur de la ligne du prix Bid	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BEAR	La couleur	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	du corps de la bougie de baissier	
CHART_COLOR_CANDLE_BULL	La couleur du corps de la bougie d'haussier	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_DOWN	La couleur de la barre en bas, de l'ombre et de l'encadrement du corps de la bougie de baissier	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_LINE	La couleur de la ligne du graphique et des chandeliers japonais "Doji"	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_UP	La couleur de la barre en haut, de l'ombre et de l'encadrement du	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	corps de la bougie d'haussier	
CHART_COLOR_FOREGROUND	La couleur des axes, les échelles et les lignes OHLC	ChartSetInteger, ChartGetInteger
CHART_COLOR_GRID	La couleur de la grille	ChartSetInteger, ChartGetInteger
CHART_COLOR_LAST	La couleur de la ligne du prix du dernier marché passé (Last)	ChartSetInteger, ChartGetInteger
CHART_COLOR_STOP_LEVEL	La couleur des niveaux des ordres Stop (Stop Loss et Take Profit)	ChartSetInteger, ChartGetInteger
CHART_COLOR_VOLUME	La couleur des volumes et des niveaux de l'ouvertu	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	re des positions	
CHART_COMMENT	Le texte du commentaire sur le graphique	ChartSetString, ChartGetString
CHART_CURRENT_POS	La position courante	ChartNavigate
CHART_DRAG_TRADE_LEVELS	L'autorisation de faire glisser les niveaux commerciaux sur le graphique à l'aide de la souris. Par défaut, le mode glisser-déposer est activé (la valeur true)	ChartSetInteger, ChartGetInteger
CHART_EVENT_MOUSE_MOVE	L'expédition des messages à tous les programmes mql5 sur	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	le graphique sur les événements du déplacement et les pressions des boutons de la souris (CHART_EVENT_MOUSE_MOVE)	
CHART_EVENT_OBJECT_CREATE	L'expédition des messages à tous les programmes mql5 sur le graphique sur l'événement de la création de l'objet graphique (CHART_EVENT_OBJECT_CREATE)	ChartSetInteger, ChartGetInteger
CHART_EVENT_OBJECT_DELETE	L'expédition des messages à tous les programmes	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	mql5 sur le graphique sur l'événement de la destruction de l'objet graphique (CHART_EVENT_OBJECT_DELETE)	
CHART_FIRST_VISIBLE_BAR	Le numéro de la première barre visible sur le graphique. L'indexation des barres correspond à la série temporelle.	ChartSetInteger, ChartGetInteger
CHART_FIXED_MAX	Le maximum fixé du graphique	ChartSetDouble, ChartGetDouble
CHART_FIXED_MIN	Le minimum fixé du graphique	ChartSetDouble, ChartGetDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
CHART_FIXED_POSITION	La situation de la position fixée du graphique du bout gauche en pourcentage. La position fixée du graphique est désignée par un petit triangle gris sur l'axe horizontal du temps et est montrée seulement dans le cas où on déconnecte le défilement automatique vers le bord droit à l'entrée du nouveau tick (regarde la propriété	ChartSetDouble, ChartGetDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	CHART_AUTOSCROLL). La barre, qui se trouve sur la position fixée, reste à la même place à l'augmentation et la réduction de l'échelle.	
CHART_FOREGROUND	Le graphique des prix dans l'arrière-plan	ChartSetInteger, ChartGetInteger
CHART_HEIGHT_IN_PIXELS	La hauteur du graphique en pixels	ChartSetInteger, ChartGetInteger
CHART_IS_OBJECT	Le critère pour l'identification de l'objet "Graphique" (OBJ_CHART) - rend true pour l'objet	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	graphique. Pour un vrai graphique la valeur est égale au false	
CHART_LINE	L'affichage comme une ligne tracée par les prix Close	ChartSetInteger
CHART_MODE	Le type de graphique (les bougies, les barres ou la ligne)	ChartSetInteger, ChartGetInteger
CHART_MOUSE_SCROLL	Le défilement du graphique par un bouton gauche de la souris à l'horizontale. Le défilement selon la verticale sera aussi accessible, si	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	dans true est établi une valeur de l'une des trois propriétés: CHART_SCALEFIX, CHART_SCALEFIX_11 ou CHART_SCALE_PT_PER_BAR	
CHART_POINTS_PER_BAR	La signification de l'échelle dans les points sur la barre	ChartSetDouble, ChartGetDouble
CHART_PRICE_MAX	Le maximum du graphique	ChartSetDouble, ChartGetDouble
CHART_PRICE_MIN	Le minimum du graphique	ChartSetDouble, ChartGetDouble
CHART_SCALE	Echelle	ChartSetInteger, ChartGetInteger
CHART_SCALE_PT_PER_BAR	Le mode de l'indication de l'échelle dans les points	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	sur la barre	
CHART_SCALEFIX	Le mode de l'échelle fixée	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX_11	Le mode de l'échelle 1:1	ChartSetInteger, ChartGetInteger
CHART_SHIFT	Le mode de l'alinéa du bord droit du graphique de prix	ChartSetInteger, ChartGetInteger
CHART_SHIFT_SIZE	Le montant de l'alinéa de la barre nulle du bord droit en pourcentage	ChartSetDouble, ChartGetDouble
CHART_SHOW_ASK_LINE	L'affichage de la signification Ask par la ligne horizontale sur le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_BID_LINE	L'affichage de la signification Bid par la	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ligne horizontale sur le graphique	
CHART_SHOW_DATE_SCALE	L'affichage de l'échelle du temps sur le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_GRID	L'affichage de la grille dans le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_LAST_LINE	L'affichage de la signification Last par la ligne horizontale sur le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_OBJECT_DESCR	Les descriptions surgissantes des objets graphiques	ChartSetInteger, ChartGetInteger
CHART_SHOW_OHLC	L'affichage dans un gauche angle supérieu	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	r des significations OHLC	
CHART_SHOW_ONE_CLICK	Отображение на графике панели быстрой торговли (опция "Торговля одним кликом")	ChartSetInteger, ChartGetInteger
CHART_SHOW_PERIOD_SEP	L'affichage des délimiteurs verticaux entre les périodes voisines	ChartSetInteger, ChartGetInteger
CHART_SHOW_PRICE_SCALE	L'affichage de l'échelle de prix sur le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_TRADE_LEVELS	L'affichage sur le graphique des niveaux commerciaux (les niveaux des positions	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ouvertes, Stop Loss, Take Profit et des ordres remis)	
CHART_SHOW_VOLUMES	L'affichage des volumes sur le graphique	ChartSetInteger, ChartGetInteger
CHART_VISIBLE_BARS	Le nombre de barres sur le graphique, accessible pour l'affichage	ChartSetInteger, ChartGetInteger
CHART_VOLUME_HIDE	Les volumes ne sont pas montrés	ChartSetInteger
CHART_VOLUME_REAL	Les volumes de commerce	ChartSetInteger
CHART_VOLUME_TICK	Les volumes de tick	ChartSetInteger
CHART_WIDTH_IN_BARS	La largeur du graphique dans	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	les barres	
CHART_WIDTH_IN_PIXELS	La largeur du graphique en pixels	ChartSetInteger, ChartGetInteger
CHART_WINDOW_HANDLE	Le handle de graphique (HWND)	ChartSetInteger, ChartGetInteger
CHART_WINDOW_IS_VISIBLE	La visibilité des sous-fenêtres	ChartSetInteger, ChartGetInteger
CHART_WINDOW_YDISTANCE	La distance en pixels selon l'axe vertical Y entre le cadre supérieur de la sous-fenêtre de l'indicateur et le cadre supérieur de la fenêtre principale du graphique. A l'arrivée des événem	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ents de la souris les coordonnées du curseur sont transmises aux coordonnées de la fenêtre principale du graphique, pendant que les coordonnées des objets graphiques dans une sous-fenêtre de l'indicateur sont spécifiées relativement un angle gauche supérieur de la sous-fenêtre. La valeur est demandée pour la transfert des	

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	coordonnées absolues du graphique principal aux coordonnées locales de la sous-fenêtre pour le travail correct avec les objets graphiques, dont les coordonnées sont spécifiées relativement un angle gauche supérieur du cadre de la sous-fenêtre.	
CHART_WINDOWS_TOTAL	Le total des fenêtres du graphique, y compris les sous-fenêtres des	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	indicateurs	
CHARTEVENT_CHART_CHANGE	Le changement des dimensions du graphique ou le changement des propriétés du graphique par le dialogue des propriétés	OnChartEvent
CHARTEVENT_CLICK	Cliquer sur le graphique	OnChartEvent
CHARTEVENT_CUSTOM	Le numéro initial de l'événement du domaine des événements d'utilisateur	OnChartEvent
CHARTEVENT_CUSTOM_LAST	Le numéro final de l'événement du domaine des événements d'utilisateur	OnChartEvent

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
CHARTEVENT_KEYDOWN	Préssion de clavier	OnChartEvent
CHARTEVENT_MOUSE_MOVE	Перемещение мыши и нажатие кнопок мыши (если для графика установлено свойство CHART_EVENT_MOUSE_MOVE=true)	OnChartEvent
CHARTEVENT_OBJECT_CHANGE	L'événement du changement des propriétés de l'objet graphique par le dialogue des propriétés	OnChartEvent
CHARTEVENT_OBJECT_CLICK	Cliquer sur un objet graphique	OnChartEvent
CHARTEVENT_OBJECT_CREATE	La création de l'objet graphique (si la	OnChartEvent

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE= true)	
CHARTEVENT_OBJECT_DELETE	L'effacement de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE= true)	OnChartEvent
CHARTEVENT_OBJECT_DRAG	Le glisser-déposer de l'objet graphique	OnChartEvent
CHARTEVENT_OBJECT_ENDEDIT	La fin de l'édition du texte dans l'objet graphique Edit	OnChartEvent
CHARTS_MAX	Le nombre maximale	Les autres constantes

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	possible des graphiques ouverts ensemble dans le terminal	
CHIKOUSPAN_LINE	La ligne Chikou Span	Lignes des indicateurs
clrAliceBlue	Alice Blue	Ensemble des couleurs Web
clrAntiqueWhite	Antique White	Ensemble des couleurs Web
clrAqua	Aqua	Ensemble des couleurs Web
clrAquamarine	Aquamarine	Ensemble des couleurs Web
clrBeige	Beige	Ensemble des couleurs Web
clrBisque	Bisque	Ensemble des couleurs Web
clrBlack	Black	Ensemble des couleurs Web
clrBlanchedAlmond	Blanched Almond	Ensemble des couleurs Web
clrBlue	Blue	Ensemble des couleurs Web
clrBlueViolet	Blue Violet	Ensemble des couleurs Web
clrBrown	Brown	Ensemble des couleurs Web
clrBurlyWood	Burly Wood	Ensemble des couleurs Web
clrCadetBlue	Cadet Blue	Ensemble des couleurs Web
clrChartreuse	Chartreuse	Ensemble des couleurs Web
clrChocolate	Chocolate	Ensemble des couleurs Web

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
clrCoral	Coral	Ensemble des couleurs Web
clrCornflowerBlue	Cornflower Blue	Ensemble des couleurs Web
clrCornsilk	Cornsilk	Ensemble des couleurs Web
clrCrimson	Crimson	Ensemble des couleurs Web
clrDarkBlue	Dark Blue	Ensemble des couleurs Web
clrDarkGoldenrod	Dark Goldenrod	Ensemble des couleurs Web
clrDarkGray	Dark Gray	Ensemble des couleurs Web
clrDarkGreen	Dark Green	Ensemble des couleurs Web
clrDarkKhaki	Dark Khaki	Ensemble des couleurs Web
clrDarkOliveGreen	Dark Olive Green	Ensemble des couleurs Web
clrDarkOrange	Dark Orange	Ensemble des couleurs Web
clrDarkOrchid	Dark Orchid	Ensemble des couleurs Web
clrDarkSalmon	Dark Salmon	Ensemble des couleurs Web
clrDarkSeaGreen	Dark Sea Green	Ensemble des couleurs Web
clrDarkSlateBlue	Dark Slate Blue	Ensemble des couleurs Web
clrDarkSlateGray	Dark Slate Gray	Ensemble des couleurs Web
clrDarkTurquoise	Dark Turquoise	Ensemble des couleurs Web

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
clrDarkViolet	Dark Violet	Ensemble des couleurs Web
clrDeepPink	Deep Pink	Ensemble des couleurs Web
clrDeepSkyBlue	Deep Sky Blue	Ensemble des couleurs Web
clrDimGray	Dim Gray	Ensemble des couleurs Web
clrDodgerBlue	Dodger Blue	Ensemble des couleurs Web
clrFireBrick	Fire Brick	Ensemble des couleurs Web
clrForestGreen	Forest Green	Ensemble des couleurs Web
clrGainsboro	Gainsboro	Ensemble des couleurs Web
clrGold	Gold	Ensemble des couleurs Web
clrGoldenrod	Goldenrod	Ensemble des couleurs Web
clrGray	Gray	Ensemble des couleurs Web
clrGreen	Green	Ensemble des couleurs Web
clrGreenYellow	Green Yellow	Ensemble des couleurs Web
clrHoneydew	Honeydew	Ensemble des couleurs Web
clrHotPink	Hot Pink	Ensemble des couleurs Web
clrIndianRed	Indian Red	Ensemble des couleurs Web
clrIndigo	Indigo	Ensemble des couleurs Web
clrIvory	Ivory	Ensemble des couleurs Web
clrKhaki	Khaki	Ensemble des couleurs Web
clrLavender	Lavender	Ensemble des couleurs Web
clrLavenderBlush	Lavender Blush	Ensemble des couleurs Web

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
clrLawnGreen	Lawn Green	Ensemble des couleurs Web
clrLemonChiffon	Lemon Chiffon	Ensemble des couleurs Web
clrLightBlue	Light Blue	Ensemble des couleurs Web
clrLightCoral	Light Coral	Ensemble des couleurs Web
clrLightCyan	Light Cyan	Ensemble des couleurs Web
clrLightGoldenrod	Light Goldenrod	Ensemble des couleurs Web
clrLightGray	Light Gray	Ensemble des couleurs Web
clrLightGreen	Light Green	Ensemble des couleurs Web
clrLightPink	Light Pink	Ensemble des couleurs Web
clrLightSalmon	Light Salmon	Ensemble des couleurs Web
clrLightSeaGreen	Light Sea Green	Ensemble des couleurs Web
clrLightSkyBlue	Light Sky Blue	Ensemble des couleurs Web
clrLightSlateGray	Light Slate Gray	Ensemble des couleurs Web
clrLightSteelBlue	Light Steel Blue	Ensemble des couleurs Web
clrLightYellow	Light Yellow	Ensemble des couleurs Web
clrLime	Lime	Ensemble des couleurs Web
clrLimeGreen	Lime Green	Ensemble des couleurs Web

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
clrLinen	Linen	Ensemble des couleurs Web
clrMagenta	Magenta	Ensemble des couleurs Web
clrMaroon	Maroon	Ensemble des couleurs Web
clrMediumAquamarine	Medium Aquamarine	Ensemble des couleurs Web
clrMediumBlue	Medium Blue	Ensemble des couleurs Web
clrMediumOrchid	Medium Orchid	Ensemble des couleurs Web
clrMediumPurple	Medium Purple	Ensemble des couleurs Web
clrMediumSeaGreen	Medium Sea Green	Ensemble des couleurs Web
clrMediumSlateBlue	Medium Slate Blue	Ensemble des couleurs Web
clrMediumSpringGreen	Medium Spring Green	Ensemble des couleurs Web
clrMediumTurquoise	Medium Turquoise	Ensemble des couleurs Web
clrMediumVioletRed	Medium Violet Red	Ensemble des couleurs Web
clrMidnightBlue	Midnight Blue	Ensemble des couleurs Web
clrMintCream	Mint Cream	Ensemble des couleurs Web
clrMistyRose	Misty Rose	Ensemble des couleurs Web
clrMoccasin	Moccasin	Ensemble des couleurs Web
clrNavajoWhite	Navajo White	Ensemble des couleurs Web

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
clrNavy	Navy	Ensemble des couleurs Web
clrNONE	L'absence de la couleur	Les autres constantes
clrOldLace	Old Lace	Ensemble des couleurs Web
clrOlive	Olive	Ensemble des couleurs Web
clrOliveDrab	Olive Drab	Ensemble des couleurs Web
clrOrange	Orange	Ensemble des couleurs Web
clrOrangeRed	Orange Red	Ensemble des couleurs Web
clrOrchid	Orchid	Ensemble des couleurs Web
clrPaleGoldenrod	Pale Goldenrod	Ensemble des couleurs Web
clrPaleGreen	Pale Green	Ensemble des couleurs Web
clrPaleTurquoise	Pale Turquoise	Ensemble des couleurs Web
clrPaleVioletRed	Pale Violet Red	Ensemble des couleurs Web
clrPapayaWhip	Papaya Whip	Ensemble des couleurs Web
clrPeachPuff	Peach Puff	Ensemble des couleurs Web
clrPeru	Peru	Ensemble des couleurs Web
clrPink	Pink	Ensemble des couleurs Web
clrPlum	Plum	Ensemble des couleurs Web
clrPowderBlue	Powder Blue	Ensemble des couleurs Web
clrPurple	Purple	Ensemble des couleurs Web
clrRed	Red	Ensemble des couleurs Web

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
clrRosyBrown	Rosy Brown	Ensemble des couleurs Web
clrRoyalBlue	Royal Blue	Ensemble des couleurs Web
clrSaddleBrown	Saddle Brown	Ensemble des couleurs Web
clrSalmon	Salmon	Ensemble des couleurs Web
clrSandyBrown	Sandy Brown	Ensemble des couleurs Web
clrSeaGreen	Sea Green	Ensemble des couleurs Web
clrSeashell	Seashell	Ensemble des couleurs Web
clrSienna	Sienna	Ensemble des couleurs Web
clrSilver	Silver	Ensemble des couleurs Web
clrSkyBlue	Sky Blue	Ensemble des couleurs Web
clrSlateBlue	Slate Blue	Ensemble des couleurs Web
clrSlateGray	Slate Gray	Ensemble des couleurs Web
clrSnow	Snow	Ensemble des couleurs Web
clrSpringGreen	Spring Green	Ensemble des couleurs Web
clrSteelBlue	Steel Blue	Ensemble des couleurs Web
clrTan	Tan	Ensemble des couleurs Web
clrTeal	Teal	Ensemble des couleurs Web
clrThistle	Thistle	Ensemble des couleurs Web
clrTomato	Tomato	Ensemble des couleurs Web
clrTurquoise	Turquoise	Ensemble des couleurs Web
clrViolet	Violet	Ensemble des couleurs Web
clrWheat	Wheat	Ensemble des couleurs Web
clrWhite	White	Ensemble des couleurs Web

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
clrWhiteSmoke	White Smoke	Ensemble des couleurs Web
clrYellow	Yellow	Ensemble des couleurs Web
clrYellowGreen	Yellow Green	Ensemble des couleurs Web
CORNER_LEFT_LOWER	Le centre des coordonnées dans un angle gauche inférieur du graphique	ObjectSetInteger, ObjectGetInteger
CORNER_LEFT_UPPER	Le centre des coordonnées dans un angle gauche supérieur du graphique	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_LOWER	Le centre des coordonnées dans un angle droit inférieur du graphique	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_UPPER	Le centre des	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	coordonnées dans un angle droit supérieur du graphique	
CP_ACP	Une page de code courant ANSI le codage au système opérationnel Windows	CharArrayToString, StringToCharArray, FileOpen
CP_MACCP	Une page de code courant Macintosh. Note: Cette signification est utilisée principalement dans les codes de programme auparavant créés et maintenant on n'a pas besoin de lui, puisque	CharArrayToString, StringToCharArray, FileOpen

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	les ordinateurs modernes Macintosh utilisent le codage Unicode.	
CP_OEMCP	Une page de code courant OEM.	CharArrayToString, StringToCharArray, FileOpen
CP_SYMBOL	La page de code Symbol	CharArrayToString, StringToCharArray, FileOpen
CP_THREAD_ACP	Le codage Windows ANSI pour le thread courant d'exécution.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF7	La page de code UTF-7.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF8	La page de code UTF-8.	CharArrayToString, StringToCharArray, FileOpen
CRYPT_AES128	Шифрование AES с ключом 128 бит (16 байт)	CryptEncode, CryptDecode
CRYPT_AES256	Шифрование	CryptEncode, CryptDecode

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	AES с ключом 256 бит (32 байта)	
CRYPT_ARCH_ZIP	ZIP архивирование	CryptEncode, CryptDecode
CRYPT_BASE64	Шифрование BASE64 (перекодировка)	CryptEncode, CryptDecode
CRYPT_DES	Шифрование DES с ключом 56 бит (7 байт)	CryptEncode, CryptDecode
CRYPT_HASH_MD5	Расчёт HASH MD5	CryptEncode, CryptDecode
CRYPT_HASH_SHA1	Расчёт HASH SHA1	CryptEncode, CryptDecode
CRYPT_HASH_SHA256	Расчёт HASH SHA256	CryptEncode, CryptDecode
DBL_DIG	La quantité de signes significants décimaux	Constantes des types de données numériques
DBL_EPSILON	La valeur minimale, à qui satisfait la	Constantes des types de données numériques

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	condition : 1.0+DBL_EPSILON != 1.0	
DBL_MANT_DIG	La quantité de bits dans la mantisse	Constantes des types de données numériques
DBL_MAX	La valeur maximale, qui peut être présentée par le type double	Constantes des types de données numériques
DBL_MAX_10_EXP	La valeur maximale décimale du degré d'exposant	Constantes des types de données numériques
DBL_MAX_EXP	La valeur maximale binaire du degré d'exposant	Constantes des types de données numériques
DBL_MIN	La valeur minimale positive, qui peut être présentée par le	Constantes des types de données numériques

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	type double	
DBL_MIN_10_EXP	La valeur minimale décimale du degré d'exposant	Constantes des types de données numériques
DBL_MIN_EXP	La valeur minimale binaire du degré d'exposant	Constantes des types de données numériques
DEAL_COMMENT	Le commentaire au le marché	HistoryDealGetString
DEAL_COMMISSION	La commission du marché	HistoryDealGetDouble
DEAL_ENTRY	La direction du marché - les entrées au marché, la sortie du marché ou le retournement	HistoryDealGetInteger
DEAL_ENTRY_IN	L'entrée au marché	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
DEAL_ENTRY_INOUT	L'inversion	HistoryDealGetInteger
DEAL_ENTRY_OUT	La sortie du marché	HistoryDealGetInteger
DEAL_MAGIC	Magic number pour la position (regardes ORDER_MAGIC)	HistoryDealGetInteger
DEAL_ORDER	L'ordre, à la base de quel le marché est effectué	HistoryDealGetInteger
DEAL_POSITION_ID	L'identificateur de la position, à l'ouverture, au changement ou à la clôture où ce marché a participé. Chaque position possède l'identificateur unique, qui est attribué à tous les	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	marchés, exécutés à l'instrument pendant toute la durée de vie de la position.	
DEAL_PRICE	Le prix du marché	HistoryDealGetDouble
DEAL_PROFIT	Le résultat financier du marché	HistoryDealGetDouble
DEAL_SWAP	Le swap accumulé à la clôture	HistoryDealGetDouble
DEAL_SYMBOL	Le nom du symbole, selon lequel le marché est produit	HistoryDealGetString
DEAL_TIME	Le temps du marché	HistoryDealGetInteger
DEAL_TIME_MSC	Le temps de l'accomplissement du marché en milliseco	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ndes depuis le 01.01.1970	
DEAL_TYPE	Le type du marché	HistoryDealGetInteger
DEAL_TYPE_BALANCE	La mise en compte de la balance	HistoryDealGetInteger
DEAL_TYPE_BONUS	L'énumération des bonus	HistoryDealGetInteger
DEAL_TYPE_BUY	L'achat	HistoryDealGetInteger
DEAL_TYPE_BUY_CANCELED	Le marché annulé de l'achat. La situation est possible, quand le marché sur l'achat auparavant fait est annulé. Dans ce cas, le type du marché auparavant fait (DEAL_TYPE_BUY) se change sur	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	DEAL_TYPE_BUY_CANCELED, et son bénéfice/perte est remise à zéro. Le bénéfice/perte auparavant reçu est calculée/annulé du compte par l'opération distincte de balance	
DEAL_TYPE_CHARGE	Les taxes supplémentaires	HistoryDealGetInteger
DEAL_TYPE_COMMISSION	Les commissions supplémentaires	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_DAILY	Агентская комиссия, начисляемая в конце торгового дня	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Агентская	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	комиссия, начисляемая в конце месяца	
DEAL_TYPE_COMMISSION_DAILY	La commission calculée à la fin du jour commercial	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_MONTHLY	La commission calculée à la fin du mois	HistoryDealGetInteger
DEAL_TYPE_CORRECTION	L'enregistrement corrigeant	HistoryDealGetInteger
DEAL_TYPE_CREDIT	La mise en compte du crédit	HistoryDealGetInteger
DEAL_TYPE_INTEREST	Начисления процентов на свободные средства	HistoryDealGetInteger
DEAL_TYPE_SELL	La vente	HistoryDealGetInteger
DEAL_TYPE_SELL_CANCELED	Le marché annulé de la vente.	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	La situation est possible, quand le marché sur la vente auparavant fait est annulé. Dans ce cas, le type du marché auparavant fait (DEAL_TYPE_SELL) se change sur DEAL_TYPE_SELL_CANCELED, et son bénéfice/perte est remise à zéro. Le bénéfice/perte auparavant reçu est calculée/annulé du compte par l'opération distincte de balance	

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
DEAL_VOLUME	Le volume du marché	HistoryDealGetDouble
DRAW_ARROW	Le dessin par les flèches	Styles du dessin
DRAW_BARS	L'affichage comme une séquence de barres	Styles du dessin
DRAW_CANDLES	L'affichage comme des bougies	Styles du dessin
DRAW_COLOR_ARROW	Le dessin par les flèches multicolores	Styles du dessin
DRAW_COLOR_BARS	Les barres multicolores	Styles du dessin
DRAW_COLOR_CANDLES	Les bougies multicolores	Styles du dessin
DRAW_COLOR_HISTOGRAM	L'histogramme multicolore de la ligne zéro	Styles du dessin
DRAW_COLOR_HISTOGRAM2	L'histogramme multicol	Styles du dessin

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ore aux deux tampons d'indicateurs	
DRAW_COLOR_LINE	La ligne multicolore	Styles du dessin
DRAW_COLOR_SECTION	Les segments multicolores	Styles du dessin
DRAW_COLOR_ZIGZAG	ZigZag multicolore	Styles du dessin
DRAW_FILLING	Le remplissage coloré entre les deux niveaux	Styles du dessin
DRAW_HISTOGRAM	L'histogramme de la ligne zéro	Styles du dessin
DRAW_HISTOGRAM2	L'histogramme des deux tampons d'indicateurs	Styles du dessin
DRAW_LINE	La ligne	Styles du dessin
DRAW_NONE	Ne dessine pas	Styles du dessin
DRAW_SECTION	Les segments	Styles du dessin

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
DRAW_ZIGZAG	Le style Zigzag admet les segments verticaux sur la barre	Styles du dessin
ELLIOTT_CYCLE	Une boucle (Cycle)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_GRAND_SUPERCYCLE	Un Supercycle principal (Grand Supercycle)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_INTERMEDIATE	Un chaînon intermédiaire (Intermediate)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINOR	Une boucle secondaire (Minor)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUETTE	Un seconde (Minuette)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUTE	Une minute (Minute)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_PRIMARY	Une boucle primaire (Primary)	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ELLIOTT_SUBMINUETTE	Un subseconde (Subminuette)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUPERCYCLE	Un Supercycle (Supercycle)	ObjectSetInteger, ObjectGetInteger
EMPTY_VALUE	La valeur vide dans le tampon d'indicateur	Les autres constantes
ERR_ACCOUNT_WRONG_PROPERTY	L'identificateur faux de la propriété du compte	GetLastError
ERR_ARRAY_BAD_SIZE	La grandeur de tableau demandé excède 2 gigoctets	GetLastError
ERR_ARRAY_RESIZE_ERROR	Il ne suffit pas la mémoire pour la redistribution du tableau ou la tentative du changem	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ent de la grandeur du tableau statique	
ERR_BOOKS_CANNOT_ADD	Le profondeur de marché ne peut pas être ajouté	GetLastError
ERR_BOOKS_CANNOT_DELETE	Le profondeur de marché ne peut pas être supprimé	GetLastError
ERR_BOOKS_CANNOT_GET	Les données du profondeur de marché ne peuvent pas être reçues	GetLastError
ERR_BOOKS_CANNOT_SUBSCRIBE	L'erreur à la souscription sur la réception des nouvelles données du profondeur de marché	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_BUFFERS_NO_MEMORY	Il ne suffit pas la mémoire pour la distribution des tampons d'indicateur	GetLastError
ERR_BUFFERS_WRONG_INDEX	L'index faux du tampon d'indicateur	GetLastError
ERR_CANNOT_CLEAN_DIRECTORY	On n'a pas réussi à vider le répertoire (probablement, un ou quelques fichiers sont bloqués et l'opération de l'effacement n'a pas réussi)	GetLastError
ERR_CANNOT_DELETE_DIRECTORY	Le répertoire ne peut pas être supprimé	GetLastError
ERR_CANNOT_DELETE_FILE	L'erreur de l'efface	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ment du fichier	
ERR_CANNOT_OPEN_FILE	L'erreur de l'ouverture du fichier	GetLastError
ERR_CHAR_ARRAY_ONLY	Le tableau doit être du type char	GetLastError
ERR_CHART_CANNOT_CHANGE	L'erreur au changement de chart du symbole et la période	GetLastError
ERR_CHART_CANNOT_CREATE_TIMER	L'erreur à la création du minuteur	GetLastError
ERR_CHART_CANNOT_OPEN	L'erreur de l'ouverture de chart	GetLastError
ERR_CHART_INDICATOR_CANNOT_ADD	L'erreur à l'ajout de l'indicateur sur le graphique	GetLastError
ERR_CHART_INDICATOR_CANNOT_DEL	L'erreur à la suppression de l'indicate	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ur du graphique	
ERR_CHART_INDICATOR_NOT_FOUND	L'indicateur n'est pas trouvé sur le graphique indiqué	GetLastError
ERR_CHART_NAVIGATE_FAILED	L'erreur de la navigation dans le chart	GetLastError
ERR_CHART_NO_EXPERT	Le chart n'a pas d'expert, qui pourrait traiter l'événement	GetLastError
ERR_CHART_NO_REPLY	Le chart ne répond pas	GetLastError
ERR_CHART_NOT_FOUND	Le chart n'est pas trouvé	GetLastError
ERR_CHART_SCREENSHOT_FAILED	L'erreur à la création des screenshots	GetLastError
ERR_CHART_TEMPLATE_FAILED	L'erreur à l'application de la modèle	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_CHART_WINDOW_NOT_FOUND	La sous-fenêtre contenant l'indicateur indiqué, n'est pas trouvé	GetLastError
ERR_CHART_WRONG_ID	L'indificateur faux de chart	GetLastError
ERR_CHART_WRONG_PARAMETER	La valeur erronée du paramètre pour la fonction du travail avec le graphique	GetLastError
ERR_CHART_WRONG_PROPERTY	L'identificateur faux de la propriété du chart	GetLastError
ERR_CUSTOM_WRONG_PROPERTY	L'identificateur faux de la propriété de l'indicateur d'utilisateur	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_DIRECTORY_NOT_EXIST	Le répertoire n'existe pas	GetLastError
ERR_DOUBLE_ARRAY_ONLY	Le tableau doit être du type double	GetLastError
ERR_FILE_BINSTRINGSIZE	La dimension de chaîne doit être spécifiée, puisque le fichier est ouvert comme binaire	GetLastError
ERR_FILE_CACHEBUFFER_ERROR	Pas assez de mémoire pour le cache pour la lecture	GetLastError
ERR_FILE_CANNOT_REWRITE	Le fichier ne peut pas être réécrit	GetLastError
ERR_FILE_IS_DIRECTORY	Ce n'est pas un fichier, c'est un répertoire	GetLastError
ERR_FILE_ISNOT_DIRECTORY	C'est un fichier, pas un	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	répertoire	
ERR_FILE_NOT_EXIST	Le fichier n'existe pas	GetLastError
ERR_FILE_NOTBIN	Le fichier doit être ouvert comme binaire	GetLastError
ERR_FILE_NOTCSV	Le fichier doit être ouvert comme CSV	GetLastError
ERR_FILE_NOTTOREAD	Le fichier doit être ouvert pour la lecture	GetLastError
ERR_FILE_NOTTOWRITE	Le fichier doit être ouvert pour l'enregistrement	GetLastError
ERR_FILE_NOTTXT	Le fichier doit être ouvert comme le texte	GetLastError
ERR_FILE_NOTTXTORCSV	Le fichier doit être ouvert comme	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	le texte ou CSV	
ERR_FILE_READERROR	L'erreur de la lecture du fichier	GetLastError
ERR_FILE_WRITEERROR	On n'a pas réussi à enregistrer la ressource au fichier	GetLastError
ERR_FLOAT_ARRAY_ONLY	Le tableau doit être du type float	GetLastError
ERR_FTP_SEND_FAILED	On ne réussit pas à expédier le fichier selon ftp	GetLastError
ERR_FUNCTION_NOT_ALLOWED	La fonction systémique n'est pas autorisé pour l'appel	GetLastError
ERR_GLOBALVARIABLE_EXISTS	La variable globale du terminal de client avec un tel nom existe déjà	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_GLOBALVARIABLE_NOT_FOUND	La variable globale du terminal de client n'est pas trouvée	GetLastError
ERR_HISTORY_NOT_FOUND	L'histoire demandée n'est pas trouvée	GetLastError
ERR_HISTORY_WRONG_PROPERTY	L'identificateur faux de la propriété de l'histoire	GetLastError
ERR_INCOMPATIBLE_ARRAYS	Le copiage des tableaux incompatibles. Le tableau de ligne peut être copié seulement à celui de ligne, et le tableau numérique - à celui numérique	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_INCOMPATIBLE_FILE	Pour les tableaux de ligne doit être un fichier de texte, pour les autres - le fichier binaire	GetLastError
ERR_INDICATOR_CANNOT_ADD	L'erreur à l' incrémentation de l'indicateur	GetLastError
ERR_INDICATOR_CANNOT_APPLY	L'indicateur ne peut pas être appliqué à un autre indicateur	GetLastError
ERR_INDICATOR_CANNOT_CREATE	L'indicateur ne peut pas être créé	GetLastError
ERR_INDICATOR_CUSTOM_NAME	Le premier paramètre dans le tableau doit être le nom de l'indicateur	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	d'utilisateur	
ERR_INDICATOR_DATA_NOT_FOUND	Les données demandées ne sont pas trouvées	GetLastError
ERR_INDICATOR_NO_MEMORY	Pas assez de mémoire pour ajouter l'indicateur	GetLastError
ERR_INDICATOR_PARAMETER_TYPE	Le type incorrect du paramètre dans le tableau à la création de l'indicateur	GetLastError
ERR_INDICATOR_PARAMETERS_MISSING	Les paramètres manquent à la création de l'indicateur	GetLastError
ERR_INDICATOR_UNKNOWN_SYMBOL	Le symbole inconnu	GetLastError
ERR_INDICATOR_WRONG_HANDLE	Ошибочный хэндл индикатора	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_INDICATOR_WRONG_INDEX	Le handle de l'indicateur erroné	GetLastError
ERR_INDICATOR_WRONG_PARAMETERS	La quantité incorrecte de paramètres à la création de l'indicateur	GetLastError
ERR_INT_ARRAY_ONLY	Le tableau doit être du type int	GetLastError
ERR_INTERNAL_ERROR	L'erreur inattendue intérieure	GetLastError
ERR_INVALID_ARRAY	Le tableau du type inconvenant, de la grandeur inconvenante ou u l'objet abîmé du tableau dynamique	GetLastError
ERR_INVALID_DATETIME	La valeur incorrecte de la	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	date et/ou le temps	
ERR_INVALID_FILEHANDLE	Le fichier avec un tel handle a été fermé, ou ne s'ouvrait pas du tout	GetLastError
ERR_INVALID_PARAMETER	Le paramètre faux à l'appel intérieur de la fonction systémique	GetLastError
ERR_INVALID_POINTER	L'indicateur faux	GetLastError
ERR_INVALID_POINTER_TYPE	Le type de l'indicateur faux	GetLastError
ERR_LONG_ARRAY_ONLY	Le tableau doit être du type long	GetLastError
ERR_MAIL_SEND_FAILED	On ne réussit pas à expédier le message	GetLastError
ERR_MARKET_LASTTIME_UNKNOWN	Le temps	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	du dernier tique est inconnu (il n'y avait pas de ticks)	
ERR_MARKET_NOT_SELECTED	Le symbole n'est pas choisi dans Market Watch	GetLastError
ERR_MARKET_SELECT_ERROR	L'erreur de l'ajout ou de la suppression du symbole dans Market Watch	GetLastError
ERR_MARKET_UNKNOWN_SYMBOL	Le symbole inconnu	GetLastError
ERR_MARKET_WRONG_PROPERTY	L'identificateur faux de la propriété du symbole	GetLastError
ERR_MQL5_WRONG_PROPERTY	L'identificateur faux de la propriété du programme	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_NO_STRING_DATE	Il n'y a pas de date dans la chaîne	GetLastError
ERR_NOT_ENOUGH_MEMORY	Il ne suffit pas la mémoire pour l'exécution de la fonction systémique	GetLastError
ERR_NOTIFICATION_SEND_FAILED	On n'a pas réussi d'envoyer la notification	GetLastError
ERR_NOTIFICATION_TOO_FREQUENT	L'expédition trop fréquente des notifications	GetLastError
ERR_NOTIFICATION_WRONG_PARAMETER	Le paramètre non valide pour envoyer une notification - on a transmis une chaîne vide dans une fonction SendNoti	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	fication( ) ou NULL	
ERR_NOTIFICATION_WRONG_SETTINGS	Les réglages incorrectes des notifications dans le terminal (ID n'est pas indiqué ou la permission n'est pas exposée)	GetLastError
ERR_NOTINITIALIZED_STRING	La ligne n'est pas initialisée	GetLastError
ERR_NUMBER_ARRAYS_ONLY	Le tableau doit être numérique	GetLastError
ERR_OBJECT_ERROR	L'erreur au fonctionnement de l'objet graphique	GetLastError
ERR_OBJECT_GETDATE_FAILED	Il est impossible de recevoir la date correspondant à	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	la valeur	
ERR_OBJECT_GETVALUE_FAILED	Il est impossible de recevoir la valeur correspondante à la date	GetLastError
ERR_OBJECT_NOT_FOUND	L'objet graphique n'est pas trouvé	GetLastError
ERR_OBJECT_WRONG_PROPERTY	L'identificateur faux de la propriété de l'objet graphique	GetLastError
ERR_ONEDIM_ARRAYS_ONLY	Le tableau doit être unidimentionnel	GetLastError
ERR_OPENCL_BUFFER_CREATE	L'erreur de la création du tampon OpenCL	GetLastError
ERR_OPENCL_CONTEXT_CREATE	L'erreur à la création du contexte OpenCL	GetLastError
ERR_OPENCL_EXECUTE	L'erreur de	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	l'exécution du programme OpenCL	
ERR_OPENCL_INTERNAL	l'erreur intérieure à l'exécution d'OpenCL	GetLastError
ERR_OPENCL_INVALID_HANDLE	Le handle incorrect OpenCL	GetLastError
ERR_OPENCL_KERNEL_CREATE	L'erreur de la création du point-kernel de l'entrée OpenCL	GetLastError
ERR_OPENCL_NOT_SUPPORTED	Les fonctions OpenCL ne sont pas soutenues sur cet ordinateur	GetLastError
ERR_OPENCL_PROGRAM_CREATE	L'erreur pendant la compilation du programme OpenCL	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_OPENCL_QUEUE_CREATE	L'erreur de la création du tour de l'exécution dans OpenCL	GetLastError
ERR_OPENCL_SET_KERNEL_PARAMETER	L'erreur à l'installation des paramètres pour le kernel OpenCL (le point de l'entrée au programme OpenCL)	GetLastError
ERR_OPENCL_TOO_LONG_KERNEL_NAME	Un trop long nom du point de l'entrée (le kernel OpenCL)	GetLastError
ERR_OPENCL_WRONG_BUFFER_OFFSET	Un décalage incorrect dans le tampon OpenCL	GetLastError
ERR_OPENCL_WRONG_BUFFER_SIZE	Une taille incorrecte du tampon OpenCL	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_PLAY_SOUND_FAILED	On ne réussit pas à reproduire le son	GetLastError
ERR_RESOURCE_NAME_DUPLICATED	a coïncidence des nomsdes ressources et dynamique etstatique	GetLastError
ERR_RESOURCE_NAME_IS_TOO_LONG	Le nom de la ressource dépasse 63 symboles	GetLastError
ERR_RESOURCE_NOT_FOUND	La ressource avec un tel nom à EX5 n'est pas trouvée	GetLastError
ERR_RESOURCE_UNSUPPORTED_TYPE	Le type non soutenu de la ressource ou la taille plus 16 Mb	GetLastError
ERR_SERIES_ARRAY	La série temporelle ne peut pas	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	être utilisée	
ERR_SHORT_ARRAY_ONLY	Le tableau doit être du type short	GetLastError
ERR_SMALL_ARRAY	Un trop petit tableau, la position de départ est à l'extérieur du tableau	GetLastError
ERR_SMALL_ASSERIES_ARRAY	Le tableau de la réception est déclaré comme AS_SERIES, et il est de la taille insuffisante	GetLastError
ERR_STRING_OUT_OF_MEMORY	Pas assez de mémoire pour la chaîne	GetLastError
ERR_STRING_RESIZE_ERROR	Il ne suffit pas la mémoire pour la redistribution de la ligne	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_STRING_SMALL_LEN	La longueur de chaîne est moins qu'attendue	GetLastError
ERR_STRING_TIME_ERROR	L'erreur de la conversion de la chaîne à la date	GetLastError
ERR_STRING_TOO_BIGNUMBER	Le nombre est trop grand, plus que ULONG_MAX	GetLastError
ERR_STRING_UNKNOWNTYPE	Le type de données inconnu à la conversion à la chaîne	GetLastError
ERR_STRING_ZEROADDED	0 est ajouté à la fin de la chaîne, l'opération est inutile	GetLastError
ERR_STRINGPOS_OUTOFRANGE	La position en dehors de la chaîne	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_STRUCT_WITHOBJECTS_ORCLASS	La structure contient les objets des lignes et/ou les tableaux dynamiques et/ou la structure avec tels objets et/ou les classes	GetLastError
ERR_SUCCESS	Операция выполнена успешно	GetLastError
ERR_TERMINAL_WRONG_PROPERTY	L'identificateur faux de la propriété du terminal	GetLastError
ERR_TOO_LONG_FILENAME	Un trop long nom du fichier	GetLastError
ERR_TOO_MANY_FILES	Ne peut pas être ouvert simultanément plus de 64 fichiers	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ERR_TOO_MANY_FORMATTERS	Plus de spécificateurs de format que des paramètres	GetLastError
ERR_TOO_MANY_PARAMETERS	Plus de paramètres que des spécificateurs de format	GetLastError
ERR_TRADE_DEAL_NOT_FOUND	Le marché n'est pas trouvé	GetLastError
ERR_TRADE_DISABLED	Le commerce pour l'expert est interdit	GetLastError
ERR_TRADE_ORDER_NOT_FOUND	L'ordre n'est pas trouvée	GetLastError
ERR_TRADE_POSITION_NOT_FOUND	La position n'est pas trouvée	GetLastError
ERR_TRADE_SEND_FAILED	On n'a pas réussi à expédier la requête commerciale	GetLastError
ERR_TRADE_WRONG_PROPERTY	L'identificateur faux de la	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	propriété du commerce	
ERR_USER_ERROR_FIRST	Par ce code commencent les erreurs, spécifiées par l'utilisateur	GetLastError
ERR_WEBREQUEST_CONNECT_FAILED	Не удалось подключиться к указанному URL	GetLastError
ERR_WEBREQUEST_INVALID_ADDRESS	URL не прошел проверку	GetLastError
ERR_WEBREQUEST_REQUEST_FAILED	Ошибка в результате выполнения HTTP запроса	GetLastError
ERR_WEBREQUEST_TIMEOUT	Превышен таймаут получения данных	GetLastError
ERR_WRONG_DIRECTORYNAME	Le nom faux du répertoire	GetLastError
ERR_WRONG_FILEHANDLE	Le handle	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	faux du fichier	
ERR_WRONG_FILENAME	Le nom inadmissible du fichier	GetLastError
ERR_WRONG_FORMATSTRING	La chaîne fausse de format	GetLastError
ERR_WRONG_INTERNAL_PARAMETER	Le paramètre faux à l'appel intérieur de la fonction du terminal de client	GetLastError
ERR_WRONG_STRING_DATE	Une date fausse dans la chaîne	GetLastError
ERR_WRONG_STRING_OBJECT	L'objet abîmé de la chaîne	GetLastError
ERR_WRONG_STRING_PARAMETER	Le paramètre abîmé du type string	GetLastError
ERR_WRONG_STRING_TIME	Un temps faux dans la chaîne	GetLastError
ERR_ZEROSIZE_ARRAY	Le tableau de la	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	longueur nulle	
FILE_ACCESS_DATE	La date du dernier accès au fichier	FileGetInteger
FILE_ANSI	Les chaînes du type ANSI (les symboles d'un byte). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())	FileOpen
FILE_BIN	Le mode binaire de la lecture-enregistrement (sans conversion de la chaîne et à la chaîne). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())	FileOpen

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
FILE_COMMON	La disposition du fichier dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()), au copiage des fichiers (FileCopy(), FileMove()) et à la vérification de l'existence des fichiers (FileIsExist())	FileOpen, FileCopy, FileMove, FileIsExist
FILE_CREATE_DATE	La date de la création	FileGetInteger
FILE_CSV	Le fichier du type csv	FileOpen

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	(Tous les éléments inscrits seront transcrits vers les chaînes du type correspondant, unicode ou ansi, et se divisent par le séparateurU). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())	
FILE_END	La réception du critère de la fin du fichier	FileGetInteger
FILE_EXISTS	La vérification de l'existence	FileGetInteger
FILE_IS_ANSI	Le fichier est ouvert comme ANSI (à	FileGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	voir FILE_ANSI)	
FILE_IS_BINARY	Le fichier est ouvert comme binaire (à voir FILE_BIN)	FileGetInteger
FILE_IS_COMMON	Le fichier est ouvert dans un dossier commun de tous les terminaux de client (à voir FILE_COMMON)	FileGetInteger
FILE_IS_CSV	Le fichier est ouvert comme CSV (à voir FILE_CSV)	FileGetInteger
FILE_IS_READABLE	Le fichier est ouvert avec la possibilité de la lecture (à voir	FileGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	FILE_READ)	
FILE_IS_TEXT	Le fichier est ouvert comme celui de texte (à voir FILE_TXT)	FileGetInteger
FILE_IS_WRITABLE	Le fichier est ouvert avec la possibilité de l'enregistrement (à voir FILE_WRITE)	FileGetInteger
FILE_LINE_END	La réception du critère de la fin de la ligne	FileGetInteger
FILE_MODIFY_DATE	La date du dernier changement	FileGetInteger
FILE_POSITION	La position du pointeur au fichier	FileGetInteger
FILE_READ	Le fichier	FileOpen

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	s'ouvre pour la lecture. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()). A l'ouverture du fichier on doit indiquer absolument le drapeau FILE_WRITE et/ou le drapeux FILE_READ.	
FILE_REWRITE	La possibilité de réécrire le fichier par les fonctions FileCopy() et FileMove(). Le fichier doit exister ou s'ouvrir pour l'enregistrement. Dans le	FileCopy, FileMove

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	cas contraire le fichier ne sera pas ouvert	
FILE_SHARE_READ	L'accès commun selon la lecture de plusieurs programmes. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()), mais ne remplace pas à l'ouverture du fichier la nécessité d'indiquer FILE_WRITE et/ou le drapeau FILE_READ	FileOpen
FILE_SHARE_WRITE	L'accès commun selon l'enregistrement de plusieurs	FileOpen

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	programmes. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()), mais ne remplace pas à l'ouverture du fichier la nécessité d'indiquer FILE_WRITE et/ou le drapeau FILE_READ	
FILE_SIZE	La taille du fichier en bytes	FileGetInteger
FILE_TXT	Un simple fichier de texte (le même csv, cependant le séparateur n'est pas pris en considération). Le	FileOpen

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	drapeau est utilisé à l'ouverture des fichiers (FileOpen())	
FILE_UNICODE	Les chaînes du type UNICODE (les symboles de deux bytes). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())	FileOpen
FILE_WRITE	Le fichier s'ouvre pour l'enregistrement. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()). A l'ouverture du fichier on doit indiquer	FileOpen

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	absolument le drapeau FILE_WRITE et/ou le drapeux FILE_READ.	
FLT_DIG	Le nombre de signes signifiants décimaux	Constantes des types de données numériques
FLT_EPSILON	La valeur minimale, qui satisfait à la condition : 1.0+FLT_EPSILON != 1.0	Constantes des types de données numériques
FLT_MANT_DIG	Le nombre de bits dans la mantisse	Constantes des types de données numériques
FLT_MAX	La valeur maximale, qui peut être présentée par le type float	Constantes des types de données numériques

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
FLT_MAX_10_EXP	La valeur maximale décimale du degré d'exposant	Constantes des types de données numériques
FLT_MAX_EXP	La valeur maximale binaire du degré d'exposant	Constantes des types de données numériques
FLT_MIN	La valeur minimale positive, qui peut être présentée par le type float	Constantes des types de données numériques
FLT_MIN_10_EXP	La valeur minimale décimale du degré d'exposant	Constantes des types de données numériques
FLT_MIN_EXP	La valeur minimale binaire du degré d'exposant	Constantes des types de données numériques
FRIDAY	Vendredi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
GANN_DOWN_TREND	La ligne correspond à la tendance descendante	ObjectSetInteger, ObjectGetInteger
GANN_UP_TREND	La ligne correspond à la tendance montante	ObjectSetInteger, ObjectGetInteger
GATORJAW_LINE	La ligne de mâchoire	Lignes des indicateurs
GATORLIPS_LINE	La ligne de lèvres	Lignes des indicateurs
GATORTEETH_LINE	La ligne de dents	Lignes des indicateurs
IDABORT	On a choisi le bouton Interrompre (Abort)	MessageBox
IDCANCEL	On a choisi le bouton Suppression (Cancel)	MessageBox
IDCONTINUE	On a choisi le bouton Continuer (Continue)	MessageBox

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
IDIGNORE	On a choisi le bouton Ignorer (Ignore)	MessageBox
IDNO	On a choisi le bouton No (No)	MessageBox
IDOK	On a choisi le bouton OK	MessageBox
IDRETRY	On a choisi le bouton Répétition (Retry)	MessageBox
IDTRYAGAIN	On a choisi le bouton Répéter (Try Again)	MessageBox
IDYES	On a choisi le bouton Oui (Yes)	MessageBox
IND_AC	Accelerator Oscillator	IndicatorCreate, IndicatorParameters
IND_AD	Accumulation/Distribution	IndicatorCreate, IndicatorParameters
IND_ADX	Average Directional Index	IndicatorCreate, IndicatorParameters

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
IND_ADXW	ADX by Welles Wilder	IndicatorCreate, IndicatorParameters
IND_ALLIGATOR	Alligator	IndicatorCreate, IndicatorParameters
IND_AMA	Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_AO	Awesome Oscillator	IndicatorCreate, IndicatorParameters
IND_ATR	Average True Range	IndicatorCreate, IndicatorParameters
IND_BANDS	Bollinger Bands®	IndicatorCreate, IndicatorParameters
IND_BEARS	Bears Power	IndicatorCreate, IndicatorParameters
IND_BULLS	Bulls Power	IndicatorCreate, IndicatorParameters
IND_BWMFI	Market Facilitation Index	IndicatorCreate, IndicatorParameters
IND_CCI	Commodity Channel Index	IndicatorCreate, IndicatorParameters
IND_CHAIKIN	Chaikin Oscillator	IndicatorCreate, IndicatorParameters
IND_CUSTOM	Custom indicator	IndicatorCreate, IndicatorParameters
IND_DEMA	Double Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_DEMARKER	DeMarker	IndicatorCreate, IndicatorParameters

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
IND_ENVELOPES	Envelopes	IndicatorCreate, IndicatorParameters
IND_FORCE	Force Index	IndicatorCreate, IndicatorParameters
IND_FRACTALS	Fractals	IndicatorCreate, IndicatorParameters
IND_FRAMA	Fractal Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_GATOR	Gator Oscillator	IndicatorCreate, IndicatorParameters
IND_ICHIMOKU	Ichimoku Kinko Hyo	IndicatorCreate, IndicatorParameters
IND_MA	Moving Average	IndicatorCreate, IndicatorParameters
IND_MACD	MACD	IndicatorCreate, IndicatorParameters
IND_MFI	Money Flow Index	IndicatorCreate, IndicatorParameters
IND_MOMENTUM	Momentum	IndicatorCreate, IndicatorParameters
IND_OBV	On Balance Volume	IndicatorCreate, IndicatorParameters
IND_OSMA	OsMA	IndicatorCreate, IndicatorParameters
IND_RSI	Relative Strength Index	IndicatorCreate, IndicatorParameters
IND_RVI	Relative Vigor Index	IndicatorCreate, IndicatorParameters
IND_SAR	Parabolic SAR	IndicatorCreate, IndicatorParameters
IND_STDDEV	Standard Deviation	IndicatorCreate, IndicatorParameters

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
IND_STOCHASTIC	Stochastic Oscillator	IndicatorCreate, IndicatorParameters
IND_TEMA	Triple Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_TRIX	Triple Exponential Moving Averages Oscillator	IndicatorCreate, IndicatorParameters
IND_VIDYA	Variable Index Dynamic Average	IndicatorCreate, IndicatorParameters
IND_VOLUMES	Volumes	IndicatorCreate, IndicatorParameters
IND_WPR	Williams' Percent Range	IndicatorCreate, IndicatorParameters
INDICATOR_CALCULATIONS	Les tampons auxiliaires pour les calculs intermédiaires	SetIndexBuffer
INDICATOR_COLOR_INDEX	Les couleurs du dessin	SetIndexBuffer
INDICATOR_DATA	Les données pour le dessin	SetIndexBuffer

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
INDICATOR_DIGITS	L'exactitude de l'affichage des valeurs de l'indicateur	IndicatorSetInteger
INDICATOR_HEIGHT	La hauteur fixée de la fenêtre personnelle de l'indicateur (la commande du préprocesseur#property indicator_height)	IndicatorSetInteger
INDICATOR_LEVELCOLOR	La couleur de la ligne du niveau	IndicatorSetInteger
INDICATOR_LEVELS	Le nombre de niveaux dans la fenêtre de l'indicateur	IndicatorSetInteger
INDICATOR_LEVELSTYLE	Le style de la ligne du niveau	IndicatorSetInteger
INDICATOR_LEVELTEXT	La descripti	IndicatorSetString

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	on du niveau	
INDICATOR_LEVELVALUE	La valeur du niveau	IndicatorSetDouble
INDICATOR_LEVELWIDTH	L'épaisseur de la ligne du niveau	IndicatorSetInteger
INDICATOR_MAXIMUM	Le maximum de la fenêtre de l'indicateur	IndicatorSetDouble
INDICATOR_MINIMUM	Le minimum de la fenêtre de l'indicateur	IndicatorSetDouble
INDICATOR_SHORTNAME	Le nom court de l'indicateur	IndicatorSetString
INT_MAX	La valeur maximale, qui peut être présentée par le type int	Constantes des types de données numériques
INT_MIN	La valeur minimale, qui peut être	Constantes des types de données numériques

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	présentée par le type int	
INVALID_HANDLE	Le handle incorrect	Les autres constantes
IS_DEBUG_MODE	Le critère du travail du programme mq5 en mode du débogage	Les autres constantes
IS_PROFILE_MODE	Le critère du travail du programme mq5 en mode du profilage	Les autres constantes
KIJUNSEN_LINE	La ligne Kijun-sen	Lignes des indicateurs
LICENSE_DEMO	La version démo du produit payant du Market Fonctionne seulement dans le testeur des	MQLInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	stratégies	
LICENSE_FREE	La version gratuite non limitée	MQLInfoInteger
LICENSE_FULL	Купленная лицензионная версия допускает не менее 5 активаций. Продавец может увеличить разрешенное число активаций	MQLInfoInteger
LICENSE_TIME	Версия с ограниченной по времени лицензией	MQLInfoInteger
LONG_MAX	La valeur maximale, qui peut être présentée par le type long	Constantes des types de données numériques

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
LONG_MIN	La valeur minimale, qui peut être présentée par le type long	Constantes des types de données numériques
LOWER_BAND	Une limite inférieure	Lignes des indicateurs
LOWER_HISTOGRAM	L'histogramme inférieur	Lignes des indicateurs
LOWER_LINE	Une ligne inférieure	Lignes des indicateurs
M_1_PI	1/pi	Constantes mathématiques
M_2_PI	2/pi	Constantes mathématiques
M_2_SQRTPI	2/sqrt(pi)	Constantes mathématiques
M_E	e	Constantes mathématiques
M_LN10	ln(10)	Constantes mathématiques
M_LN2	ln(2)	Constantes mathématiques
M_LOG10E	log10(e)	Constantes mathématiques
M_LOG2E	log2(e)	Constantes mathématiques
M_PI	pi	Constantes mathématiques
M_PI_2	pi/2	Constantes mathématiques
M_PI_4	pi/4	Constantes mathématiques
M_SQRT1_2	1/sqrt(2)	Constantes mathématiques
M_SQRT2	sqrt(2)	Constantes mathématiques

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
MAIN_LINE	La ligne principale	Lignes des indicateurs
MB_ABORTRETRYIGNORE	La fenêtre du message contient trois boutons: Abort, Retry et Ignore	MessageBox
MB_CANCELTRYCONTINUE	La fenêtre du message contient trois boutons: Cancel, Try Again, Continue	MessageBox
MB_DEFBUTTON1	Le premier bouton MB_DEFBUTTON1 - le bouton est choisi par défaut, si MB_DEFBUTTON2, MB_DEFBUTTON3, ou MB_DEFBUTTON4 ne	MessageBox

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	sont pas définis	
MB_DEFBUTTON2	Le deuxième bouton - le bouton par défaut	MessageBox
MB_DEFBUTTON3	Le troisième bouton - le bouton par défaut	MessageBox
MB_DEFBUTTON4	Le quatrième bouton - le bouton par défaut	MessageBox
MB_ICONEXCLAMATION, MB_ICONWARNING	L'icône de signe de question	MessageBox
MB_ICONINFORMATION, MB_ICONASTERISK	L'icône, comprenant le signe de ligne i dans le cercle	MessageBox
MB_ICONQUESTION	L'icône de signe de question	MessageBox
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	L'icône de signe STOP	MessageBox

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
MB_OK	La fenêtre du message contient un bouton: OK. Par défaut	MessageBox
MB_OKCANCEL	La fenêtre du message contient deux boutons: OK et Cancel	MessageBox
MB_RETRYCANCEL	La fenêtre du message contient deux boutons: Retry et Cancel	MessageBox
MB_YESNO	La fenêtre du message contient deux boutons: Yes et No	MessageBox
MB_YESNOCANCEL	La fenêtre du message contient trois boutons: Yes, No	MessageBox

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	et Cancel	
MINUSDI_LINE	La ligne -DI	Lignes des indicateurs
MODE_EMA	La prise de moyenne exponentielle	Méthodes des glissantes
MODE_LWMA	La prise de moyenne de ligne-pesée	Méthodes des glissantes
MODE_SMA	La prise de moyenne simple	Méthodes des glissantes
MODE_SMMA	La prise de moyenne lissée	Méthodes des glissantes
MONDAY	Lundi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
MQL_DEBUG	L'indice du travail du programme exécuté au mode du débogage	MQLInfoInteger
MQL_DLLS_ALLOWED	La permission de l'utilisation DLL pour ce program	MQLInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	me exécuté	
MQL_FRAME_MODE	Le critère du travail de l'expert lancé sur le graphique enmode du rassemblement des trames des résultats de l'optimisation	MQLInfoInteger
MQL_LICENSE_TYPE	Le type de la licence du module EX5. La licence se rapporte notamment à ce module EX5, d'où on fait la demande à l'aide de MQLInfoInteger(MQL_LICENSE_TYPE).	MQLInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
MQL_MEMORY_LIMIT	La quantité maximale possible de mémoire dynamique pour le programme MQL5 en Mb	MQLInfoInteger
MQL_MEMORY_USED	La taille de la mémoire utilisée par le programme MQL5 en Mb	MQLInfoInteger
MQL_OPTIMIZATION	L'indice du travail du programme exécuté en train de l'optimisation	MQLInfoInteger
MQL_PROFILER	Le critère du travail du programme lancé en mode du profilage du code	MQLInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
MQL_PROGRAM_NAME	Le nom du programme mql5 exécuté	MQLInfoString
MQL_PROGRAM_PATH	Le chemin pour ce programme exécuté	MQLInfoString
MQL_PROGRAM_TYPE	Le type de programme mql5	MQLInfoInteger
MQL_SIGNALS_ALLOWED	Разрешение на работу с сигналами данной запущенной программы	MQLInfoInteger
MQL_TESTER	L'indicatif du travail du programme exécuté au testeur	MQLInfoInteger
MQL_TRADE_ALLOWED	La permission du commerce pour ce programme exécuté	MQLInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
MQL_VISUAL_MODE	L'indice du travail du programme exécuté dans le mode visuel de test	MQLInfoInteger
NULL	Le zéro de n'importe quel type	Les autres constantes
OBJ_ALL_PERIODS	L'objet se dessine sur tous les temps trames	ObjectSetInteger, ObjectGetInteger
OBJ_ARROW	L'objet "La flèche"	Types des objets
OBJ_ARROW_BUY	Le signe "Buy"	Types des objets
OBJ_ARROW_CHECK	Le signe "Vérification"	Types des objets
OBJ_ARROW_DOWN	Le signe "la flèche en bas"	Types des objets
OBJ_ARROW_LEFT_PRICE	La balise gauche de prix	Types des objets
OBJ_ARROW_RIGHT_PRICE	La balise droite de prix	Types des objets

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
OBJ_ARROW_SELL	Le signe "Sell"	Types des objets
OBJ_ARROW_STOP	Le signe "Stop"	Types des objets
OBJ_ARROW_THUMB_DOWN	Le signe "mal" (le pouce en bas)	Types des objets
OBJ_ARROW_THUMB_UP	Le signe "bien" (le pouce en haut)	Types des objets
OBJ_ARROW_UP	Le signe "la flèche en haut"	Types des objets
OBJ_ARROWED_LINE	L'objet "Line avec une flèche"	Types des objets
OBJ_BITMAP	L'objet "Le dessin"	Types des objets
OBJ_BITMAP_LABEL	L'objet "La balise graphique"	Types des objets
OBJ_BUTTON	L'objet "Le bouton"	Types des objets
OBJ_CHANNEL	Le canal équidistant	Types des objets
OBJ_CHART	L'objet "Le graphique"	Types des objets
OBJ_CYCLES	Les lignes cycliques	Types des objets

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
OBJ_EDIT	L'objet "Le champ de l'entrée"	Types des objets
OBJ_ELLIOTWAVE3	Vague d'Elliott corrective	Types des objets
OBJ_ELLIOTWAVE5	Vague d'Elliott pulsatoire	Types des objets
OBJ_ELLIPSE	L'ellipse	Types des objets
OBJ_EVENT	L'objet "Événement", correspondant à l'événement dans le calendrier économique	Types des objets
OBJ_EXPANSION	L'expansion de Fibonacci	Types des objets
OBJ_FIBO	Les niveaux de Fibonacci	Types des objets
OBJ_FIBOARC	Les arcs de Fibonacci	Types des objets
OBJ_FIBOCHANNEL	Le canale de	Types des objets

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	Fibonacci	
OBJ_FIBOFAN	Le fanion de Fibonacci	Types des objets
OBJ_FIBOTIMES	Les zones temporaires de Fibonacci	Types des objets
OBJ_GANNFAN	Le fanion de Gann	Types des objets
OBJ_GANNGRID	La grille de Gann	Types des objets
OBJ_GANNLINE	La ligne de Gann	Types des objets
OBJ_HLINE	Ligne horizontale	Types des objets
OBJ_LABEL	L'objet "La balise de texte"	Types des objets
OBJ_NO_PERIODS	Объект не показывается ни на одном таймфрейме	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_D1	L'objet se dessine sur les graphiques de jour	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
OBJ_PERIOD_H1	L'objet se dessine sur les graphiques de 1 heure	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H12	L'objet se dessine sur les graphiques de 12 heures	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H2	L'objet se dessine sur les graphiques de 2 heures	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H3	L'objet se dessine sur les graphiques de 3 heures	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H4	L'objet se dessine sur les graphiques de 4 heures	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H6	L'objet se dessine sur les graphiques de 6 heures	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H8	L'objet se	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	dessine sur les graphiques de 8 heures	
OBJ_PERIOD_M1	L'objet se dessine sur les graphiques de 1 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M10	L'objet se dessine sur les graphiques de 10 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M12	L'objet se dessine sur les graphiques de 12 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M15	L'objet se dessine sur les graphiques de 15 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M2	L'objet se dessine sur les graphiques de 2 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M20	L'objet se dessine sur les	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	graphiques de 20 minutes	
OBJ_PERIOD_M3	L'objet se dessine sur les graphiques de 3 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M30	L'objet se dessine sur les graphiques de 30 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M4	L'objet se dessine sur les graphiques de 4 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M5	L'objet se dessine sur les graphiques de 5 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M6	L'objet se dessine sur les graphiques de 6 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_MN1	L'objet se dessine sur les graphiqu	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	es des mois	
OBJ_PERIOD_W1	L'objet se dessine sur les graphiques d'une semaine	ObjectSetInteger, ObjectGetInteger
OBJ_PITCHFORK	La fourche d'Andrews	Types des objets
OBJ_RECTANGLE	Le rectangle	Types des objets
OBJ_RECTANGLE_LABEL	L'objet "Marque rectangulaire" pour la création et la présentation de l'interface d'utilisateur graphique.	Types des objets
OBJ_REGRESSION	Le canale de la régression linéaire	Types des objets
OBJ_STDDEVCHANNEL	Le canal de l' écart type	Types des objets
OBJ_TEXT	L'objet "Le	Types des objets

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	texte"	
OBJ_TREND	La ligne de Tendance	Types des objets
OBJ_TRENDBYANGLE	La ligne de tendance par l'angle	Types des objets
OBJ_TRIANGLE	Le triangle	Types des objets
OBJ_VLINE	La ligne verticale	Types des objets
OBJPROP_ALIGN	L'alignement du texte à l'horizontale dans l'objet "Champ de l'entrée" (OBJ_EDIT)	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANCHOR	La position du point du rattachement de l'objet graphique	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANGLE	L'angle. Pour les objets créés du programme avec l'angle qui ne	ObjectSetDouble, ObjectGetDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	pas encore spécifié, la valeur est égale à EMPTY_VALUE	
OBJPROP_ARROWCODE	Le code de la flèche pour l'objet "La flèche"	ObjectSetInteger, ObjectGetInteger
OBJPROP_BACK	L'objet à l'arrière-plan	ObjectSetInteger, ObjectGetInteger
OBJPROP_BGCOLOR	La couleur du fond pour OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	ObjectSetInteger, ObjectGetInteger
OBJPROP_BMPFILE	Le nom du fichier BMP pour l'objet "Une balise graphique". Voir aussi Ressources	ObjectSetString, ObjectGetString
OBJPROP_BORDER_COLOR	La couleur du cadre	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	pour l'objet OBJ_EDIT et OBJ_BUTTON	
OBJPROP_BORDER_TYPE	Le type du cadre pour l'objet "Le cadre rectangulaire"	ObjectSetInteger, ObjectGetInteger
OBJPROP_CHART_ID	L'identificateur de l'objet "Graphique" (OBJ_CHART). Permet de travailler avec les propriétés de cet objet comme avec le graphique ordinaire à l'aide des fonctions du paragraphe Les opérations avec les graphiqu	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	es, mais il y a certains exceptions.	
OBJPROP_CHART_SCALE	L'échelle pour l'objet "le graphique"	ObjectSetInteger, ObjectGetInteger
OBJPROP_COLOR	La couleur	ObjectSetInteger, ObjectGetInteger
OBJPROP_CORNER	L'angle du graphique pour le rattachement de l'objet graphique	ObjectSetInteger, ObjectGetInteger
OBJPROP_CREATETIME	Le temps de la création de l'objet	ObjectSetInteger, ObjectGetInteger
OBJPROP_DATE_SCALE	Affichage de l'échelle du temps pour l'objet "Graphique"	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEGREE	Le niveau du marquage d'onde d'Elliott	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
OBJPROP_DEVIATION	La déviation pour le canal de la déviation standard	ObjectSetDouble, ObjectGetDouble
OBJPROP_DIRECTION	La tendance de l'objet de Gann	ObjectSetInteger, ObjectGetInteger
OBJPROP_DRAWLINES	L'affichage des lignes pour le marquage d'onde d'Elliott	ObjectSetInteger, ObjectGetInteger
OBJPROP_ELLIPSE	L'affichage de l'ellipse complète pour l'objet "l'Arc de Fibonacci" (OBJ_FIBOARC)	ObjectSetInteger, ObjectGetInteger
OBJPROP_FILL	Le remplissage d'un objet avec la couleur (pour OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CH	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	
OBJPROP_FONT	La fonte	ObjectSetString, ObjectGetString
OBJPROP_FONTSIZE	La dimension de la fonte	ObjectSetInteger, ObjectGetInteger
OBJPROP_HIDDEN	L'interdiction de montrer le nom d'un objet graphique dans la liste des objets du menu du terminal "Graphiques" - "Objets" - "Liste des objets". La valeur true permet de cacher l'objet inutile pour l'utilisateur de la liste. Par défaut	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	true est défini pour les objets, qui affichent les événements du calendrier, l'histoire du commerce, ainsi que pour les objets créés du programme. Pour voir tels objets graphiques et recevoir l'accès à leurs propriétés, il faut appuyer sur le bouton "Tout" dans la fenêtre "Liste des objets".	
OBJPROP_LEVELCOLOR	La couleur de la ligne-niveau	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
OBJPROP_LEVELS	Le nombre de niveaux	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELSTYLE	Le style de la ligne-niveau	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELTEXT	La description du niveau	ObjectSetString, ObjectGetString
OBJPROP_LEVELVALUE	La valeur du niveau	ObjectSetDouble, ObjectGetDouble
OBJPROP_LEVELWIDTH	L'épaisseur de la ligne-niveau	ObjectSetInteger, ObjectGetInteger
OBJPROP_NAME	Le nom de l'objet	ObjectSetString, ObjectGetString
OBJPROP_PERIOD	La période pour l'objet "le graphique"	ObjectSetInteger, ObjectGetInteger
OBJPROP_PRICE	La coordonnée du prix	ObjectSetDouble, ObjectGetDouble
OBJPROP_PRICE_SCALE	Affichage de l'échelle de prix pour l'objet "Graphique"	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
OBJPROP_RAY	Une ligne verticale passe par toutes les fenêtres d'un graphique	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_LEFT	Le rayon va à gauche	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_RIGHT	Le rayon va à droite	ObjectSetInteger, ObjectGetInteger
OBJPROP_READONLY	La possibilité de l'édition du texte dans l'objet Edit	ObjectSetInteger, ObjectGetInteger
OBJPROP_SCALE	L'échelle (la propriété des objets de Gann et l'objet "les arcs de Fibonacci")	ObjectSetDouble, ObjectGetDouble
OBJPROP_SELECTABLE	L'accessibilité de l'objet	ObjectSetInteger, ObjectGetInteger
OBJPROP_SELECTED	L'objet est sélectionné	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
OBJPROP_STATE	L'état du bouton (est appuyé/ est pressé)	ObjectSetInteger, ObjectGetInteger
OBJPROP_STYLE	Le style	ObjectSetInteger, ObjectGetInteger
OBJPROP_SYMBOL	Le caractère pour l'objet "le graphique"	ObjectSetString, ObjectGetString
OBJPROP_TEXT	La description de l'objet (le texte qui se trouve dans l'objet)	ObjectSetString, ObjectGetString
OBJPROP_TIME	La coordonnée du temps	ObjectSetInteger, ObjectGetInteger
OBJPROP_TIMEFRAMES	La visibilité de l'objet sur les temps trames	ObjectSetInteger, ObjectGetInteger
OBJPROP_TOOLTIP	Le texte de l'infobulle émergeante. Si la propriété n'est pas	ObjectSetString, ObjectGetString

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	spécifiée, se montre l'infobulle automatiquement formée par le terminal. On peut désactiver l'infobulle, en définissant sa valeur "\n" (saut de la ligne)	
OBJPROP_TYPE	Le type de l'objet	ObjectSetInteger, ObjectGetInteger
OBJPROP_WIDTH	L'épaisseur de la ligne	ObjectSetInteger, ObjectGetInteger
OBJPROP_XDISTANCE	La distance en pixels selon l'axe X de l'angle du rattachement (см. примечание)	ObjectSetInteger, ObjectGetInteger
OBJPROP_XOFFSET	La coordonnée X de l'angle	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	supérieur gauche du domaine rectangulaire de la visibilité dans les objets graphiques "Marque graphique" et "Dessin" (OBJ_BITMAP_LABEL et OBJ_BITMAP). La valeur est spécifiée en pixels par rapport à l'angle supérieur gauche de l'image initiale.	
OBJPROP_XSIZE	La largeur de l'objet selon l'axe X en pixels. Est spécifiée pour les	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	objets OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_YDISTANCE	La distance en pixels selon l'axe Y de l'angle du rattachement (см. примечание)	ObjectSetInteger, ObjectGetInteger
OBJPROP_YOFFSET	La coordonnée Y de l'angle supérieur gauche du domaine rectangulaire de la visibilité dans les objets graphiqu	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	es "Marque graphique" et "Dessin" (OBJ_BITMAP_LABEL et OBJ_BITMAP). La valeur est spécifiée en pixels par rapport à l'angle supérieur gauche de l'image initiale.	
OBJPROP_YSIZE	La hauteur de l'objet selon l'axe Y en pixels. Est spécifiée pour les objets OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BIT	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	MAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_ZORDER	La priorité de l'objet graphique sur la réception de l'événement de clic de la souris sur le graphique (CHART_EVENT_CLICK). Par défaut la valeur est égal au zéro à la création, mais la priorité peut être augmentée en cas de nécessité. En appliquant des objets un sur un autre,	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	seulement un objet avec la plus haute priorité recevra l'événement CHARTEVENT_CLICK.	
ORDER_COMMENT	Le commentaire	OrderGetString, HistoryOrderGetString
ORDER_FILLING_FOK	Cette politique de l'exécution signifie que l'ordre peut être exceptionnellement exécuté dans le volume indiqué. Si au marché au moment donné il n'y a pas de volume suffisant de l'instrument financier	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	, l'ordre ne sera pas exécuté. Le volume nécessaire peut être fait de quelques propositions accessibles au moment donné sur le marché.	
ORDER_FILLING_IOC	Signifie l'accord de passer le marché par le volume au maximum accessible sur le marché indiqué à l'ordre. En cas d'impossibilité de l'exécution complète l'ordre sera exécuté sur le volume accessible, et le	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	volume non exécuté de l'ordre sera annulé.	
ORDER_FILLING_RETURN	Ce mode est utilisé pour les ordres de marché(ORDER_TYPE_BUY et ORDER_TYPE_SELL), limites et de stop de limite (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT) et seulement en modes "Exécution selon le marché"	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	et "Exécution de bourse". En cas de l'exécution partielle l'ordre de marché ou limité avec le volume résiduel n'est pas remis et continue de fonctionner. Un ordre correspondant limité ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT avec le type de l'exécution ORDER_FILLING_RETURN sera créé à l'activation pour les ordres ORDER_TYPE_BU	

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	Y_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT.	
ORDER_MAGIC	L'identificateur de l'expert exposant l'ordre (est destiné pour que chaque expert expose le numéro personnel unique)	OrderGetInteger, HistoryOrderGetInteger
ORDER_POSITION_ID	L'identificateur de la position, qui est mis sur l'ordre à son exécution. Chaque ordre exécuté engendre le marché, qui ouvreune nouvelle postion ou change	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	la position déjà existante. L'identificateur de cette position s'établit à l'ordre exécuté à ce moment.	
ORDER_PRICE_CURRENT	Le prix actuel selon le symbole de l'ordre	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_OPEN	Le prix indiqué dans l'ordre	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_STOPLIMIT	Le prix de l'organisation de l'ordre à cours limité au fonctionnement de l'ordre StopLimit	OrderGetDouble, HistoryOrderGetDouble
ORDER_SL	Le niveau Stop Loss	OrderGetDouble, HistoryOrderGetDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ORDER_STATE	Le statut de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_CANCELED	L'ordre est retiré par le client	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_EXPIRED	L'ordre est retiré à l'expiration du terme ses actions	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_FILLED	L'ordre a été exécuté complètement	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PARTIAL	L'ordre a été exécuté partiellement	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PLACED	L'ordre est accepté	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REJECTED	L'ordre est rejeté	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_ADD	L'ordre dans l'état de l'enregistrement (l'exposition au système commercial)	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ORDER_STATE_REQUEST_CANCEL	L'ordre dans l'état de la suppression (la suppression du système commercial)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_MODIFY	L'ordre dans l'état de la modification (le changement de ses paramètres)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_STARTED	L'ordre est contrôlé sur la convenance, mais n'est pas encore accepté par le broker	OrderGetInteger, HistoryOrderGetInteger
ORDER_SYMBOL	Le symbole, selon lequel on expose l'ordre	OrderGetString, HistoryOrderGetString
ORDER_TIME_DAY	L'ordre fonctionnera seulement	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	pendant le jour commercial courant	
ORDER_TIME_DONE	Le temps de l'exécution ou le retrait de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE_MSC	Le temps de l'exécution/du retrait de l'ordre en millisecondes depuis le 01.01.1970	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_EXPIRATION	Le temps de l'expiration de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_GTC	L'ordre se trouve au tour jusqu'à ce qu'il soit retiré	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP	Le temps de l'organis	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ation de l'ordre	
ORDER_TIME_SETUP_MSC	Le temps de l'établissement de l'ordre pour l'exécution en millisecondes depuis le 01.01.1970	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED	L'ordre fonctionnera jusqu'à la date de l'expiration	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED_DAY	L'ordre sera valable jusqu'à 23:59:59 du jour indiqué. Si ce temps ne tombe pas sur la session commerciale, l'expiration se produira au temps	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	commercial immédiat.	
ORDER_TP	Le niveau Take Profit	OrderGetDouble, HistoryOrderGetDouble
ORDER_TYPE	Le type de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY	L'ordre de marché sur l'achat	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_LIMIT	L'ordre remis Buy Limit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP	L'ordre remis Buy Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP_LIMIT	Pour l'obtention du prix de l'ordre se présente l'ordre remis Buy Limit au prix de StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_FILLING	Le type de l'exécution selon le reste	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
ORDER_TYPE_SELL	L'ordre de marché sur la vente	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_LIMIT	L'ordre remis Sell Limit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP	L'ordre remis Sell Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP_LIMIT	Pour l'obtention du prix de l'ordre se présente l'ordre remis Sell Limit au prix de StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_TIME	Le temps de la vie de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_VOLUME_CURRENT	Le volume non exécuté	OrderGetDouble, HistoryOrderGetDouble
ORDER_VOLUME_INITIAL	Le volume initial à l'organisation de l'ordre	OrderGetDouble, HistoryOrderGetDouble
PERIOD_CURRENT	Période actuelle	Périodes des graphiques

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
PERIOD_D1	1 jour	Périodes des graphiques
PERIOD_H1	1 heure	Périodes des graphiques
PERIOD_H12	12 heures	Périodes des graphiques
PERIOD_H2	2 heures	Périodes des graphiques
PERIOD_H3	3 heures	Périodes des graphiques
PERIOD_H4	4 heures	Périodes des graphiques
PERIOD_H6	6 heures	Périodes des graphiques
PERIOD_H8	8 heures	Périodes des graphiques
PERIOD_M1	1 minute	Périodes des graphiques
PERIOD_M10	10 minutes	Périodes des graphiques
PERIOD_M12	12 minutes	Périodes des graphiques
PERIOD_M15	15 minutes	Périodes des graphiques
PERIOD_M2	2 minutes	Périodes des graphiques
PERIOD_M20	20 minutes	Périodes des graphiques
PERIOD_M3	3 minutes	Périodes des graphiques
PERIOD_M30	30 minutes	Périodes des graphiques
PERIOD_M4	4 minutes	Périodes des graphiques
PERIOD_M5	5 minutes	Périodes des graphiques
PERIOD_M6	6 minutes	Périodes des graphiques

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
PERIOD_MN1	1 mois	Périodes des graphiques
PERIOD_W1	1 semaine	Périodes des graphiques
PLOT_ARROW	Le code de la flèche pour le style DRAW_ARROW	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_ARROW_SHIFT	Le décalage des flèches selon la verticale pour le style DRAW_ARROW	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_COLOR_INDEXES	Le nombre de couleurs	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_BEGIN	Le nombre de barres initialles sans dessin et les valeurs dans DataWindow	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_TYPE	Le type de la construction graphique	PlotIndexSetInteger, PlotIndexGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
PLOT_EMPTY_VALUE	Une valeur vide pour la construction, pour laquelle il n'y a aucun dessin	PlotIndexSetDouble
PLOT_LABEL	Le nom de la série d'indicateur graphique pour l'affichage dans la fenêtre DataWindow. Pour les styles complexes graphiques demandant pour l'affichage quelques tampons d'indicateur, on peut spécifier les noms pour chaque tampon en utilisant comme	PlotIndexSetString

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	délimiteur ";". L'exemple du code est dans DRAW_CANDLES	
PLOT_LINE_COLOR	L'index du tampon qui contient la couleur du dessin	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_STYLE	Le style de la ligne du dessin	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_WIDTH	L'épaisseur de la ligne du dessin	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHIFT	Le décalage de la construction graphique de l'indicateur selon l'axe du temps dans les barres	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHOW_DATA	Le critère de l'affichage des valeurs de la	PlotIndexSetInteger, PlotIndexGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	construction dans la fenêtre DataWindow	
PLUSDI_LINE	La ligne +DI	Lignes des indicateurs
POINTER_AUTOMATIC	L'indicateur de n'importe quel objet créé automatiquement (sans utilisation new())	CheckPointer
POINTER_DYNAMIC	L'indicateur de l'objet créé par l'opérateur new	CheckPointer
POINTER_INVALID	L'indicateur incorrect	CheckPointer
POSITION_COMMENT	Le commentaire sur la position	PositionGetString
POSITION_COMMISSION	La commission	PositionGetDouble
POSITION_IDENTIFIER	L'identificateur de la position est un nombre unique,	PositionGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	qui est assigné à chaque position ouverte à nouveau, et ne change pas pendant toute sa vie. Le viremen t de la position ne change pas l'identificateur de la position.	
POSITION_MAGIC	Magic number pour la position (regardes ORDER_MAGIC)	PositionGetInteger
POSITION_PRICE_CURRENT	La valeur présente selon le symbole	PositionGetDouble
POSITION_PRICE_OPEN	Le prix de la position	PositionGetDouble
POSITION_PROFIT	Un profit courant	PositionGetDouble
POSITION_SL	Le niveau Stop Loss	PositionGetDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	pour une position ouverte	
POSITION_SWAP	Le swap accumulé	PositionGetDouble
POSITION_SYMBOL	Le symbole selon lequel la position est ouverte	PositionGetString
POSITION_TIME	Le temps de l'ouverture de la position	PositionGetInteger
POSITION_TIME_MSC	Le temps de l'ouverture de la position en millisecondes depuis le 01.01.1970	PositionGetInteger
POSITION_TIME_UPDATE	Le temps du changement de la position en millisecondes depuis le 01.01.1970	PositionGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
POSITION_TIME_UPDATE_MSC	Le temps du changement de la position en millisecondes depuis le 01.01.1970	PositionGetInteger
POSITION_TP	Le niveau Take Profit pour une position ouverte	PositionGetDouble
POSITION_TYPE	Le type de la position	PositionGetInteger
POSITION_TYPE_BUY	L'achat	PositionGetInteger
POSITION_TYPE_SELL	La vente	PositionGetInteger
POSITION_VOLUME	Le volume de la position	PositionGetDouble
PRICE_CLOSE	Le prix de la clôture	Constantes de prix
PRICE_HIGH	Le prix maximum pour la période	Constantes de prix
PRICE_LOW	Le prix minimum pour la période	Constantes de prix

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
PRICE_MEDIAN	Le prix moyen, (high+low)/2	Constantes de prix
PRICE_OPEN	Le prix de l'ouverture	Constantes de prix
PRICE_TYPICAL	Le prix typique, (high+low+close)/3	Constantes de prix
PRICE_WEIGHTED	Le prix moyen pondéré, (high+low+close+close)/4	Constantes de prix
PROGRAM_EXPERT	L'expert	MQLInfoInteger
PROGRAM_INDICATOR	L'indicateur	MQLInfoInteger
PROGRAM_SCRIPT	Le script	MQLInfoInteger
REASON_ACCOUNT	Un autre compte a été activé ou la reconnexion vers le serveur commercial est produite à la suite de changements dans les paramèt	UninitializeReason, OnDeinit

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	res du compte	
REASON_CHARTCHANGE	Le caractère ou la période du graphique était changé	UninitializeReason, OnDeinit
REASON_CHARTCLOSE	Le graphique est fermé	UninitializeReason, OnDeinit
REASON_CLOSE	Le terminal a été fermé	UninitializeReason, OnDeinit
REASON_INITFAILED	Cette valeur signifit que OnInit() a rendu la valeur non nulle	UninitializeReason, OnDeinit
REASON_PARAMETERS	Les paramètres d'entrée étaient changés par l'utilisateur	UninitializeReason, OnDeinit
REASON_PROGRAM	L'expert a cessé le travail, ayant provoqué la fonction	UninitializeReason, OnDeinit

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ExpertRemove()	
REASON_RECOMPILE	Le programme a été recompilé	UninitializeReason, OnDeinit
REASON_REMOVE	Le programme est supprimé du graphique	UninitializeReason, OnDeinit
REASON_TEMPLATE	Un autre modèle du graphique a été appliqué	UninitializeReason, OnDeinit
SATURDAY	Samedi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SEEK_CUR	La position courante de l'indicateur de fichier	FileSeek
SEEK_END	La fin du fichier	FileSeek
SEEK_SET	Le début du fichier	FileSeek
SENKOUSPANA_LINE	La ligne Senkou Span A	Lignes des indicateurs
SENKOUSPANB_LINE	La ligne Senkou Span B	Lignes des indicateurs

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
SERIES_BARS_COUNT	Le nombre de barres selon le symbole-la période pour ce moment	SeriesInfoInteger
SERIES_FIRSTDATE	La première date pour le symbole-la période pour le moment actuel	SeriesInfoInteger
SERIES_LASTBAR_DATE	Le temps de l'ouverture de la dernière barre selon le caractère-période	SeriesInfoInteger
SERIES_SERVER_FIRSTDATE	La première date à l'histoire selon le symbole sur le serveur indépendamment de la période	SeriesInfoInteger
SERIES_SYNCHRONIZED	Признак синхрон	SeriesInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	изированности данных по символу/периоду на данный момент	
SERIES_TERMINAL_FIRSTDATE	La première date à l'histoire selon le symbole dans le terminal de client indépendamment de la période	SeriesInfoInteger
SHORT_MAX	La valeur maximale, qui peut être présentée par le type short	Constantes des types de données numériques
SHORT_MIN	La valeur minimale, qui peut être présentée par le type short	Constantes des types de données numériques
SIGNAL_BASE_AUTHOR_LOGIN	Логин автора сигнала	SignalBaseGetString

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
SIGNAL_BASE_BALANCE	Баланс счета	SignalBaseGetDouble
SIGNAL_BASE_BROKER	Наименование брокера (компании)	SignalBaseGetString
SIGNAL_BASE_BROKER_SERVER	Сервер брокера	SignalBaseGetString
SIGNAL_BASE_CURRENCY	Валюта счета сигнала	SignalBaseGetString
SIGNAL_BASE_DATE_PUBLISHED	Дата публикации сигнала (когда стал доступен для подписки)	SignalBaseGetInteger
SIGNAL_BASE_DATE_STARTED	Дата начала мониторинга сигнала	SignalBaseGetInteger
SIGNAL_BASE_EQUITY	Средства на счете	SignalBaseGetDouble
SIGNAL_BASE_GAIN	Прирост счета в процентах	SignalBaseGetDouble
SIGNAL_BASE_ID	ID сигнала	SignalBaseGetInteger
SIGNAL_BASE_LEVERAGE	Плечо торгового счета	SignalBaseGetInteger
SIGNAL_BASE_MAX_DRAWDOWN	Максимальная	SignalBaseGetDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	просадка	
SIGNAL_BASE_NAME	Имя сигнала	SignalBaseGetString
SIGNAL_BASE_PIPS	Результат торговли в пипсах	SignalBaseGetInteger
SIGNAL_BASE_PRICE	Цена подписки на сигнал	SignalBaseGetDouble
SIGNAL_BASE_RATING	Позиция в рейтинге сигналов	SignalBaseGetInteger
SIGNAL_BASE_ROI	Значение ROI (Return on Investment) сигнала в %	SignalBaseGetDouble
SIGNAL_BASE_SUBSCRIBERS	Количество подписчиков	SignalBaseGetInteger
SIGNAL_BASE_TRADE_MODE	Тип счета (0-реальный счет, 1-демо-счет, 2-конкурсный счет)	SignalBaseGetInteger
SIGNAL_BASE_TRADES	Количество	SignalBaseGetInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	трейдов	
SIGNAL_INFO_CONFIRMATIONS_DISABLED	Флаг разрешения синхронизации без показа диалога подтверждения	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_COPY_SLTP	Флаг копирования Stop Loss и Take Profit	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_DEPOSIT_PERCENT	Ограничения по депозиту (в %)	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_EQUITY_LIMIT	Процент для конвертации объема сделки	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_ID	id сигнала, r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_NAME	Имя сигнала, r/o	SignalInfoGetString
SIGNAL_INFO_SLIPPAGE	Величина проскальзывания, с которым выставляются рыночн	SignalInfoGetDouble, SignalInfoSetDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ые ордера при синхронизации позиций и копировании сделок	
SIGNAL_INFO_SUBSCRIPTION_ENABLED	Флаг разрешения на копирование сделок по подписке	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_TERMS_AGREE	Флаг согласия с условиями использования сервиса "Сигналы", r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_VOLUME_PERCENT	Значение ограничения по средствам для сигнала, r/o	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_LINE	La ligne de signal	Lignes des indicateurs
STAT_BALANCE_DD	Un prélèvement maximal de la balance	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	en moyens monétaires. En train du commerce la balance peut éprouver la multitude de prélèvements, il faut prendre la plus grande signification.	
STAT_BALANCE_DD_RELATIVE	Le prélèvement de la balance en moyens monétaires, qui a été fixé au moment du prélèvement maximal de la balance en pourcentage (STAT_BALANCE_DDREL_PERCENT).	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
STAT_BALANCE_DDREL_PERCENT	Un prélèvement maximal de la balance en pourcentage. En train du commerce la balance peut éprouver la multitude de prélèvements, on fixe une valeur relative en pourcentage et une valeur absolue monétaire pour chaque prélèvement. Rend la plus grande valeur	TesterStatistics
STAT_BALANCEDD_PERCENT	Le prélèvement de la balance en pourcentage, qui	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	a été fixé au moment du prélèvement maximal de la balance en moyens monétaires (STAT_BALANCE_DD).	
STAT_BALANCEMIN	La valeur minimale de la balance	TesterStatistics
STAT_CONLOSSMAX	Une perte maximale au héritage des trades déficitaires. La valeur est moins ou égal au zéro	TesterStatistics
STAT_CONLOSSMAX_TRADES	Le nombre de trades qui ont formé STAT_CONLOSSMAX (une perte maximal	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	e au héritage des trades déficitaires)	
STAT_CONPROFITMAX	Un bénéfice maximal dans la plus grande valeur parmi toutes les trades profitables. La valeur est plus ou égal au zéro	TesterStatistics
STAT_CONPROFITMAX_TRADES	Le nombre de trades qui ont formé STAT_CONPROFITMAX (un bénéfice maximal au héritage des trades profitables)	TesterStatistics
STAT_CUSTOM_ONTESTER	La valeur du critère calculé d'utilisat	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	eur de l'optimisation rendu par la fonction OnTester()	
STAT_DEALS	Le nombre de marchés faits	TesterStatistics
STAT_EQUITY_DD	Un prélèvement maximal des moyens en moyens monétaires. En train du commerce les moyens peuvent éprouver la multitude de prélèvements, il faut prendre la plus grande signification.	TesterStatistics
STAT_EQUITY_DD_RELATIVE	Le prélèvement des moyens en	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	moyens monétaires, qui a été fixé au moment du prélèvement maximal des moyens en pourcentage (STAT_EQUITY_DDREL_PERCENT).	
STAT_EQUITY_DDREL_PERCENT	Un prélèvement maximal des moyens en pourcentage. En train du commerce les moyens peuvent éprouver la multitude de prélèvements, dont on fixe une valeur relative du prélèvement en porcenta	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ge. Rend la plus grande valeur	
STAT_EQUITYDD_PERCENT	>Le prélèvement des moyens en pourcentage, qui a été fixé au moment du prélèvement maximal des moyens en moyens monétaires (STAT_EQUITY_DD).	TesterStatistics
STAT_EQUITYMIN	La valeur minimale des moyens propres	TesterStatistics
STAT_EXPECTED_PAYOFF	L'attente mathématique du gain	TesterStatistics
STAT_GROSS_LOSS	Une perte totale, la somme de tous les	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	trades déficitaires (négatifs). La valeur est moins ou égal au zéro	
STAT_GROSS_PROFIT	Le bénéfice total, la somme de tous les trades profitables (positifs). La valeur est plus ou égal au zéro	TesterStatistics
STAT_INITIAL_DEPOSIT	La valeur du dépôt initial	TesterStatistics
STAT_LONG_TRADES	Les trades longues	TesterStatistics
STAT_LOSS_TRADES	Les trades déficitaires	TesterStatistics
STAT_LOSSTRADES_AVGCON	Une moyenne longueur de la série déficitaire des trades	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
STAT_MAX_CONLOSS_TRADES	Le nombre de trades dans la plus longue série de trades déficitaresSTAT_MAX_CONLOSSES	TesterStatistics
STAT_MAX_CONLOSSES	La perte totale dans une plus longue série des trades déficitaires	TesterStatistics
STAT_MAX_CONPROFIT_TRADES	Le nombre de trades dans la plus longue série des trades profitablesSTAT_MAX_CONWINS	TesterStatistics
STAT_MAX_CONWINS	Le bénéfice total dans la plus longue série des	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	trades profitables	
STAT_MAX_LOSSTRADE	Une perte maximale - la valeur la plus petite parmi tous les trades déficitaires. La valeur est moins ou égal au zéro	TesterStatistics
STAT_MAX_PROFITTRADE	Un bénéfice maximal - la plus grande valeur parmi tous les trades profitables. La valeur est plus ou égal au zéro	TesterStatistics
STAT_MIN_MARGINLEVEL	La valeur minimale atteinte du niveau de la marge	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
STAT_PROFIT	Le bénéfice net après le test, la somme STAT_GROSS_PROFIT et STAT_GROSS_LOSS (STAT_GROSS_LOSS est toujours moins ou égal au zéro)	TesterStatistics
STAT_PROFIT_FACTOR	La rentabilité - le rapportSTAT_GROSS_PROFIT/STAT_GROSS_LOSS. Si STAT_GROSS_LOSS=0, la rentabilité prend la valeurDBL_MAX	TesterStatistics
STAT_PROFIT_LONGTRADES	Les trades longues profitables	TesterStatistics
STAT_PROFIT_SHORTTRADES	Les trades courts	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	profitables	
STAT_PROFIT_TRADES	Les trades profitables	TesterStatistics
STAT_PROFITTRADES_AVGCON	Une moyenne longueur de la série profitable des trades	TesterStatistics
STAT_RECOVERY_FACTOR	Le facteur de la restitution - le rapportSTAT_PROFIT/STAT_BALANCE_DD	TesterStatistics
STAT_SHARPE_RATIO	Le coefficient de Scharp	TesterStatistics
STAT_SHORT_TRADES	Les trades courts	TesterStatistics
STAT_TRADES	Le nombre de trades	TesterStatistics
STAT_WITHDRAWAL	Le nombre de moyens déduits du compte	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
STO_CLOSECLOSE	La construction aux prix Close/Close	Constantes de prix
STO_LOWHIGH	La construction aux prix Low/High	Constantes de prix
STYLE_DASH	La ligne interrompue	Styles du dessin
STYLE_DASHDOT	La barre- la ligne pointillée	Styles du dessin
STYLE_DASHDOTDOT	La barre-deux points	Styles du dessin
STYLE_DOT	La ligne pointillée	Styles du dessin
STYLE_SOLID	La ligne continue	Styles du dessin
SUNDAY	Dimanche	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SYMBOL_ASK	Ask - une meilleure proposition pour l'achat	SymbolInfoDouble
SYMBOL_ASKHIGH	Ask maximu	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	m par jour	
SYMBOL_ASKLOW	Ask minimum par jour	SymbolInfoDouble
SYMBOL_BANK	La source de la cotation en cours	SymbolInfoString
SYMBOL_BASIS	Имя базового актива для производного инструмента	SymbolInfoString
SYMBOL_BID	Bid - une meilleure proposition pour la vente	SymbolInfoDouble
SYMBOL_BIDHIGH	Bid maximum par jour	SymbolInfoDouble
SYMBOL_BIDLOW	Bid minimum par jour	SymbolInfoDouble
SYMBOL_CALC_MODE_CFD	CFD mode - le calcul du gage et du profit pour CFD	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDINDEX	CFD index	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	mode - le calcul du gage et du profit pour CFD par les index	
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD Leverage mode - le calcul du gage et du profit pour CFD au commerce d'effet de levier	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES	Futures mode - le calcul de la garantie et des bénéfices pour la commerce des contrats à terme sur la bourse	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS Futures mode - le calcul de la garantie et des bénéfices pour la commer	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ce des contrats à terme sur FORTS.	
SYMBOL_CALC_MODE_EXCH_STOCKS	Exchange mode - le calcul de la garantie et des bénéfices pour la commerce des papiers monétaires sur la bourse	SymbolInfoInteger
SYMBOL_CALC_MODE_FOREX	Forex mode - le calcul du profit et la marge pour Forex	SymbolInfoInteger
SYMBOL_CALC_MODE_FUTURES	Futures mode - le calcul du gage et du profit pour les futures	SymbolInfoInteger
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - a symbol is used as a non-tradable asset on a	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	trading account.	
SYMBOL_CURRENCY_BASE	La devise de base de l'instrument	SymbolInfoString
SYMBOL_CURRENCY_MARGIN	La devise des moyens hypothécaires	SymbolInfoString
SYMBOL_CURRENCY_PROFIT	La devise du bénéfice	SymbolInfoString
SYMBOL_DESCRIPTION	La description de ligne du caractère	SymbolInfoString
SYMBOL_DIGITS	Le nombre de signes après la virgule	SymbolInfoInteger
SYMBOL_EXPIRATION_DAY	L'ordre est valable jusqu'à la fin du jour	SymbolInfoInteger
SYMBOL_EXPIRATION_GTC	L'ordre est valable sans limitation par le temps	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	jusqu'à son annulation évidente	
SYMBOL_EXPIRATION_MODE	Les drapeaux des modes permis de l'expiration de l'ordre	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED	Le délai de l'expiration est indiqué dans l'ordre	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED_DAY	Le jour de l'expiration est indiqué dans l'ordre	SymbolInfoInteger
SYMBOL_EXPIRATION_TIME	La date de l'achèvement des enchères selon l'instrument (d'habitude il est utilisé pour les contrats à terme)	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
SYMBOL_FILLING_FOK	Cette politique de l'exécution signifie que l'ordre peut être exceptionnellement exécuté dans le volume indiqué. Si au marché au moment donné il n'y a pas de volume suffisant de l'instrument financier, l'ordre ne sera pas exécuté. Le volume nécessaire peut être fait de quelques propositions accessibles au moment donné	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	sur le marché.	
SYMBOL_FILLING_IOC	Dans ce cas le trader accepte de passer le marché par le volume au maximum accessible sur le marché dans la limite du volume indiqué dans l'ordre. En cas d'impossibilité de l'exécution complète l'ordre sera exécuté sur le volume accessible, et le volume non exécuté de l'ordre sera annulé. La possibilité de l'utilisati	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	on des ordres IOC est définie sur le serveur commercial.	
SYMBOL_FILLING_MODE	Les drapeaux des modes permis du remplissage de l'ordre	SymbolInfoInteger
SYMBOL_ISIN	Le nom du symbole commercial dans le système des codes internationaux identificatoires des valeurs mobilières— ISIN (International Securities Identification Number). Le numéro international d'identifi	SymbolInfoString

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	cation de la valeur mobilière - c'est un code alphanumérique à 12 chiffres qui identifie la valeur mobilière absolument. La présence de la propriété donnée du symbole est définie au côté du serveur commercial.	
SYMBOL_LAST	Prix de la dernière affaire	SymbolInfoDouble
SYMBOL_LASTHIGH	Last maximum par jour	SymbolInfoDouble
SYMBOL_LASTLOW	Last minimum par jour	SymbolInfoDouble
SYMBOL_MARGIN_INITIAL	La marge	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	initiale (initiante) désigne le montant des moyens nécessaires hypothécaires en devise de marge pour l'ouverture de la position par le volume qui correspond à un lot. Elle est utilisée au contrôle des moyens du client à l'entrée au marché.	
SYMBOL_MARGIN_MAINTENANCE	Le marge de maintenance selon l'instrument. En cas si elle est spécifié	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	e - on indique le montant de la marge en devise de marge de l'instrument qui est retenue d'un lot. Elle est utilisé au contrôle des moyens du client au changement de l'état du compte du client. Si la marge de maintenance est égale au 0, on utilise la marge initiale.	
SYMBOL_OPTION_MODE	Тип опциона	SymbolInfoInteger
SYMBOL_OPTION_MODE_AMERICAN	Европейский тип опциона - может быть	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	погашен только в указанную дату (дата истечения срока, дата исполнения, дата погашения)	
SYMBOL_OPTION_MODE_EUROPEAN	Американский тип опциона - может быть погашен в любой день до истечения срока опциона. Для такого типа задаётся период, в течение которого покупатель может исполнить данный опцион	SymbolInfoInteger
SYMBOL_OPTION_RIGHT	Право опциона	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	(Call/Put)	
SYMBOL_OPTION_RIGHT_CALL	Опцион, дающий право купить актив по фиксированной цене	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_PUT	Опцион, дающий право продать актив по фиксированной цене	SymbolInfoInteger
SYMBOL_OPTION_STRIKE	Цена исполнения опциона. Это цена, по которой покупатель опциона может купить (при опционе Call) или продать (при опционе Put) базовый актив, а продавец опциона соответственно обязан продать	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	или купить соответствующее количество базового актива.	
SYMBOL_ORDER_LIMIT	Les ordres limites sont permis (Buy Limit et Sell Limit)	SymbolInfoInteger
SYMBOL_ORDER_MARKET	Les ordres de marché sont permis (Buy et Sell)	SymbolInfoInteger
SYMBOL_ORDER_MODE	Les drapeaux des types permis de l'ordre	SymbolInfoInteger
SYMBOL_ORDER_SL	L'établissement de Stop Loss est permis	SymbolInfoInteger
SYMBOL_ORDER_STOP	Les ordres stop sont permis (Buy Stop et	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	Sell Stop)	
SYMBOL_ORDER_STOP_LIMIT	Les ordres stop limites sont permis (Buy Stop Limit et Sell Stop Limit)	SymbolInfoInteger
SYMBOL_ORDER_TP	L'établissement de Take Profit est permis	SymbolInfoInteger
SYMBOL_PATH	Le chemin dans l'arbre des caractères	SymbolInfoString
SYMBOL_POINT	La valeur de point de symbole	SymbolInfoDouble
SYMBOL_SELECT	Le symbole est choisi dans Market Watch	SymbolInfoInteger
SYMBOL_SESSION_AW	Le prix moyen pondéré de la session	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
SYMBOL_SESSION_BUY_ORDERS	Le nombre total d'ordres à l'achat pour le moment	SymbolInfoInteger
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Le volume total des ordres à l'achat pour le moment	SymbolInfoDouble
SYMBOL_SESSION_CLOSE	Le prix de la clôture de la session	SymbolInfoDouble
SYMBOL_SESSION_DEALS	Le nombre de marchés dans la session courante	SymbolInfoInteger
SYMBOL_SESSION_INTEREST	Le volume total des positions ouvertes	SymbolInfoDouble
SYMBOL_SESSION_OPEN	Le prix de l'ouverture de la session	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MAX	La valeur au maximum admissible du	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	prix de la session	
SYMBOL_SESSION_PRICE_LIMIT_MIN	La valeur au minimum admissible du prix de la session	SymbolInfoDouble
SYMBOL_SESSION_PRICE_SETTLEMENT	Le prix de la livraison à la session courante	SymbolInfoDouble
SYMBOL_SESSION_SELL_ORDERS	Le nombre total d'ordres à la vente pour le moment	SymbolInfoInteger
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Le volume total des ordres à la vente pour le moment	SymbolInfoDouble
SYMBOL_SESSION_TURNOVER	Le chiffre d'affaires total des marchés à la session courante	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
SYMBOL_SESSION_VOLUME	Le volume total des marchés à la session courante	SymbolInfoDouble
SYMBOL_SPREAD	La valeur de spread dans les points	SymbolInfoInteger
SYMBOL_SPREAD_FLOAT	L'indication du spread flottant	SymbolInfoInteger
SYMBOL_START_TIME	La date du commencement des enchères selon l'instrument (d'habitude il est utilisé pour les contrats à terme)	SymbolInfoInteger
SYMBOL_SWAP_LONG	La valeur de swap à l'achat	SymbolInfoDouble
SYMBOL_SWAP_MODE	Le modèle de calcul de swap	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Les swaps sont calculés	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	dans l'argent en devise du dépôt du client	
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Les swaps sont calculés dans l'argent en devise de marge du symbole	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Les swaps sont calculés dans l'argent en devise de base du symbole	SymbolInfoInteger
SYMBOL_SWAP_MODE_DISABLED	Pas de swap	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Les swaps sont calculés dans les pour-cent annuels du prix de l'instrument au moment du calcul	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	du swap (le mode bancaire - 360 jours en année)	
SYMBOL_SWAP_MODE_INTEREST_OPEN	Les swaps sont calculés dans les pour-cent annuels du prix de la position de l'ouverture selon le symbole (le mode bancaire - 360 jours en année)	SymbolInfoInteger
SYMBOL_SWAP_MODE_POINTS	Les swaps sont calculés dans les points	SymbolInfoInteger
SYMBOL_SWAP_MODE_REOPEN_BID	Les swaps sont calculés par l'ouverture répétée de la position. A la fin du jour commer	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	cial la position se ferme forcément. Le lendemain la position s'ouvre de nouveau selon le prix courant Bid +/- le nombre indiqué de points (dans les paramètres SYMBOL_SWAP_LONG et SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Les swaps sont calculés par l'ouverture répétée de la position. A la fin du jour commercial la position se ferme forcément. Le	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	lendemain la position de l'ouverture répétée selon le prix de la clôture + / - le nombre indiqué de points (dans les paramètres SYMBOL_SWAP_LONG et SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_ROLLOVER3DAYS	Le jour de la semaine pour le calcul de swap triple	SymbolInfoInteger
SYMBOL_SWAP_SHORT	La valeur de swap à la vente	SymbolInfoDouble
SYMBOL_TICKS_BOOKDEPTH	Le nombre maximal des demandes montrées au profonde	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ur de marché. Pour les instruments qui n'ont pas de tour des demandes, la valeur est égal au 0	
SYMBOL_TIME	Le temps de la dernière cotation	SymbolInfoInteger
SYMBOL_TRADE_CALC_MODE	La mode de calcul des prix de contrat	SymbolInfoInteger
SYMBOL_TRADE_CONTRACT_SIZE	Le montant du contrat commercial	SymbolInfoDouble
SYMBOL_TRADE_EXECUTION_EXCHANGE	L'exécution boursière	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_INSTANT	Le commerce sur prix de courant	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_MARKET	L'exécution des ordres selon le marché	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
SYMBOL_TRADE_EXECUTION_REQUEST	Le commerce sur la demande	SymbolInfoInteger
SYMBOL_TRADE_EXEMODE	La mode d'exécution d'affaire	SymbolInfoInteger
SYMBOL_TRADE_FREEZE_LEVEL	La distance pour bloquer des opérations commerciales (dans les points)	SymbolInfoInteger
SYMBOL_TRADE_MODE	Le type d'exécution d'ordre	SymbolInfoInteger
SYMBOL_TRADE_MODE_CLOSEONLY	On permet seulement les opérations de la clôture des positions	SymbolInfoInteger
SYMBOL_TRADE_MODE_DISABLED	Le commerce selon le symbole est interdit	SymbolInfoInteger
SYMBOL_TRADE_MODE_FULL	Il n'y a pas de limitatio	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	ns sur les transactions commerciales	
SYMBOL_TRADE_MODE_LONGONLY	On permet seulement les achats	SymbolInfoInteger
SYMBOL_TRADE_MODE_SHORTONLY	On permet seulement les ventes	SymbolInfoInteger
SYMBOL_TRADE_STOPS_LEVEL	L'alinéa minimal dans les points du prix courant de la clôture pour l'installation des ordres Stop	SymbolInfoInteger
SYMBOL_TRADE_TICK_SIZE	Le changement minimal du prix	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE	La valeur SYMBOL_TRADE_TICK_VALUE_PROFIT	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_LOSS	Le coût calculé du tick	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	pour la position déficitaire	
SYMBOL_TRADE_TICK_VALUE_PROFIT	Le coût calculé du tick pour la position profitable	SymbolInfoDouble
SYMBOL_VOLUME	Le volume - le volume dans le dernier marché	SymbolInfoInteger
SYMBOL_VOLUME_LIMIT	Un volume cumulé au maximum admissible pour le symbole donné de la position ouverte et des ordres remis dans une direction (l'achat ou la vente). Par exemple, à la limitation aux 5 lots on	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	peut avoir la position ouverte sur l'achat par le volume de 5 lots et exposer l'ordre remis Sell Limit par le volume de 5 lots. Mais on ne peut pas exposer l'ordre remis Buy Limit (puisque un volume cumulé dans une direction excéder a la limitation) ou exposer Sell Limit par le volume plus de 5 lots.	
SYMBOL_VOLUME_MAX	Le volume maximal	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	pour une affaire	
SYMBOL_VOLUME_MIN	Le volume minimal pour une affaire	SymbolInfoDouble
SYMBOL_VOLUME_STEP	Un pas minimal du changement du volume pour une affaire	SymbolInfoDouble
SYMBOL_VOLUMEHIGH	Le volume maximum par jour	SymbolInfoInteger
SYMBOL_VOLUMELOW	Le volume minimum par jour	SymbolInfoInteger
TENKANSEN_LINE	La ligne Tenkan-sen	Lignes des indicateurs
TERMINAL_BUILD	Le numéro de build du terminal lancé	TerminalInfoInteger
TERMINAL_CODEPAGE	Le numéro de la page de code du langage est mis dans le	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	terminal de client	
TERMINAL_COMMONDATA_PATH	Le dossier commun pour tous les terminals de client, installés sur un ordinateur	TerminalInfoString
TERMINAL_COMMUNITY_ACCOUNT	Флаг наличия авторизационных данных MQL5.community в терминале	TerminalInfoInteger
TERMINAL_COMMUNITY_BALANCE	Баланс пользователя в MQL5.community	TerminalInfoDouble
TERMINAL_COMMUNITY_CONNECTION	Наличие подключения к MQL5.community	TerminalInfoInteger
TERMINAL_COMPANY	Le nom de la compagnie	TerminalInfoString
TERMINAL_CONNECTED	La présence de la connexion au	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	serveur commercial	
TERMINAL_CPU_CORES	Le nombre de processeurs dans le système	TerminalInfoInteger
TERMINAL_DATA_PATH	Le dossier où se trouvent les données du terminal	TerminalInfoString
TERMINAL_DISK_SPACE	La quantité d'espace disque libre pour le dossier MQL5\Files du terminal (agent) en Mb	TerminalInfoInteger
TERMINAL_DLLS_ALLOWED	La permission de l'utilisation DLL	TerminalInfoInteger
TERMINAL_EMAIL_ENABLED	La permission de l'expédition des lettres avec l'utilisation du serveur	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	SMTP et le nom d'utilisateur, indiqué dans les réglages du terminal	
TERMINAL_FTP_ENABLED	La permission de l'expédition des rapports par FTP sur le serveur indiqué pour le terminal indiqué dans les réglages du compte commercial	TerminalInfoInteger
TERMINAL_LANGUAGE	Langage du terminal	TerminalInfoString
TERMINAL_MAXBARS	Le nombre maximum des barres sur le graphique	TerminalInfoInteger
TERMINAL_MEMORY_AVAILABLE	La taille de la mémoire libre du procès du	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	terminal (agent) en Mb	
TERMINAL_MEMORY_PHYSICAL	La taille de la mémoire physique dans le système, en Mb	TerminalInfoInteger
TERMINAL_MEMORY_TOTAL	La taille de la mémoire disponible au procès du terminal (agent), en Mb	TerminalInfoInteger
TERMINAL_MEMORY_USED	La taille de la mémoire utilisée par le terminal (agent) en Mb	TerminalInfoInteger
TERMINAL_MQID	Флаг наличия MetaQuotes ID для отправки Push-уведомлений	TerminalInfoInteger
TERMINAL_NAME	Le nom du terminal	TerminalInfoString
TERMINAL_NOTIFICATIONS_ENABLED	Разрешение на отправку	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	уведомлений на смартфон	
TERMINAL_OPENCL_SUPPORT	La version de l'OpenCL soutenu dans l'aspect 0x00010002 = 1.2. "0" signifit, que OpenCL n'est pas soutenu	TerminalInfoInteger
TERMINAL_PATH	Le dossier d'où le terminal a été démarré	TerminalInfoString
TERMINAL_PING_LAST	La dernière valeur connue du ping au serveur de trading en micro-secondes. Une seconde équivaut à un million de micro-	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	secondes.	
TERMINAL_SCREEN_DPI	La résolution d'affichage de l'écran est mesurée en nombre de Points Par Pouce (PPP ou Dots Per Inch - DPI)	TerminalInfoInteger
TERMINAL_TRADE_ALLOWED	La permission au commerce	TerminalInfoInteger
TERMINAL_X64	Le critère "le terminal de 64 bits"	TerminalInfoInteger
THURSDAY	Jeudi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TRADE_ACTION_DEAL	Placer l'ordre commercial pour l'exécution immédiate avec les paramètres	MqlTradeRequest

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	indiqués (l'ordre du marché)	
TRADE_ACTION_MODIFY	Modifiez les paramètres de l'ordre placé auparavant	MqlTradeRequest
TRADE_ACTION_PENDING	Placer l'ordre commercial pour l'exécution sous les conditions indiquées (l'ordre remis)	MqlTradeRequest
TRADE_ACTION_REMOVE	Supprimer l'ordrecommercial auparavant remis	MqlTradeRequest
TRADE_ACTION_SLTP	Changer les valeurs Stop Loss et Take Profit de la position ouverte	MqlTradeRequest
TRADE_RETCODE_CANCEL	La requête est	MqlTradeResult

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	annulé par un broker	
TRADE_RETCODE_CLIENT_DISABLES_AT	L'autotrading est interdit par le terminal de client	MqlTradeResult
TRADE_RETCODE_CONNECTION	Il n'y a pas de connection avec le serveur commercial.	MqlTradeResult
TRADE_RETCODE_DONE	La requête est accomplie	MqlTradeResult
TRADE_RETCODE_DONE_PARTIAL	La requête est partiellement accomplie	MqlTradeResult
TRADE_RETCODE_ERROR	L'erreur de traitement de requête	MqlTradeResult
TRADE_RETCODE_FROZEN	L'ordre ou la position se sont gelés	MqlTradeResult
TRADE_RETCODE_INVALID	La requête incorrecte	MqlTradeResult

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
TRADE_RETCODE_INVALID_EXPIRATION	La date incorrecte de l'expiration de l'ordre dans la requête	MqlTradeResult
TRADE_RETCODE_INVALID_FILL	On indique le type non soutenu de l'exécution de l'ordre selon le reste	MqlTradeResult
TRADE_RETCODE_INVALID_ORDER	le type de l'ordre incorrecte ou interdit	MqlTradeResult
TRADE_RETCODE_INVALID_PRICE	Le prix incorrect dans la requête	MqlTradeResult
TRADE_RETCODE_INVALID_STOPS	Les Stops incorrects dans la requête	MqlTradeResult
TRADE_RETCODE_INVALID_VOLUME	Le volume incorrect dans la requête	MqlTradeResult
TRADE_RETCODE_LIMIT_ORDERS	La limite sur la quantité d'ordres remis	MqlTradeResult

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	est atteint	
TRADE_RETCODE_LIMIT_VOLUME	La limite sur le volume des ordres et les positions pour le symbole donné est atteint	MqlTradeResult
TRADE_RETCODE_LOCKED	La requête est bloquée pour le traitement	MqlTradeResult
TRADE_RETCODE_MARKET_CLOSED	Le marché est fermé	MqlTradeResult
TRADE_RETCODE_NO_CHANGES	Il n'y a pas de changements dans la requête	MqlTradeResult
TRADE_RETCODE_NO_MONEY	Il n'y a pas de ressources suffisantes pour l'exécution de la requête	MqlTradeResult
TRADE_RETCODE_ONLY_REAL	L'opération est permise seuleme	MqlTradeResult

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	nt pour les comptes réels	
TRADE_RETCODE_ORDER_CHANGED	L'état de l'ordre a changé	MqlTradeResult
TRADE_RETCODE_PLACED	L'ordre a placé	MqlTradeResult
TRADE_RETCODE_POSITION_CLOSED	La position avecPOSITION_IDENTIFIERindiquée est déjà fermée	MqlTradeResult
TRADE_RETCODE_PRICE_CHANGED	Les prix ont changé	MqlTradeResult
TRADE_RETCODE_PRICE_OFF	Les cotations pour le traitement de la requête manquent	MqlTradeResult
TRADE_RETCODE_REJECT	La requête rejetée	MqlTradeResult
TRADE_RETCODE_REQUOTE	La recitation	MqlTradeResult
TRADE_RETCODE_SERVER_DISABLES_AT	L'autotrading est interdit par le serveur	MqlTradeResult
TRADE_RETCODE_TIMEOUT	La requête	MqlTradeResult

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	est supprimée à l'expiration du temps	
TRADE_RETCODE_TOO_MANY_REQUESTS	Les requêtes trop fréquentes	MqlTradeResult
TRADE_RETCODE_TRADE_DISABLED	Le commerce est interdit	MqlTradeResult
TRADE_TRANSACTION_DEAL_ADD	L'ajout du marché dans l'histoire. Se réalise à la suite de l'exécution de l'ordre ou de la tenue des opérations avec la balance du compte.	MqlTradeTransaction
TRADE_TRANSACTION_DEAL_DELETE	La suppression du marché de l'histoire. Les situations sont	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	possibles, quand le marché auparavant exécuté est supprimé sur le serveur. Par exemple, le marché était supprimé dans le système (la bourse) extérieur commercial, où il était déduit par le broker.	
TRADE_TRANSACTION_DEAL_UPDATE	Le changement du marché dans l'histoire. Les situations sont possibles, quand le marché auparavant exécuté se change	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	sur le serveur. Par exemple, le marché était changé dans le système (la bourse) extérieur commercial, où il était déduit par le broker.	
TRADE_TRANSACTION_HISTORY_ADD	L'ajout de l'ordre dans l'histoire à la suite de l'exécution ou de la suppression.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_DELETE	La suppression de l'ordre de l'histoire des ordres. Ce type est prévu pour l'élargissement	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	de la fonctionnalité sur le côté du serveur commercial.	
TRADE_TRANSACTION_HISTORY_UPDATE	Le changement de l'ordre qui se trouve dans l'histoire des ordres. Ce type est prévu pour l'élargissement de la fonctionnalité sur le côté du serveur commercial.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_ADD	L'ajout d'un nouvel ordre ouvert.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_DELETE	La suppression de l'ordre de la liste des ouverts. L'ordre peut	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	être supprimé des ouverts à la suite de l'exposition de la demande correspondante ou à la suite de l'exécution (du remplissage) et le transfert dans l'histoire.	
TRADE_TRANSACTION_ORDER_UPDATE	Le changement de l'ordre ouvert. Non seulement les changements évidents du côté du terminal de client ou du serveur commercial se rapportent aux changements donnés,	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	mais aussi le changement de son état à son exposition (par exemple, le passage de l'étatORDER_STATE_STARTED au ORDER_STATE_PLACED ou de ORDER_STATE_PLACED au ORDER_STATE_PARTIAL etc.).	
TRADE_TRANSACTION_POSITION	Le changement de la position non lié à l'exécution du marché. Ce type de la transaction témoigne notamment de ce que la position	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	était changée sur le côté du serveur commercial. Le volume, le prix de l'ouverture, ainsi que les niveaux Stop Loss et Take Profit peuvent être changés dans la position. L'information sur les changements est transmise dans la structure MqlTradeTransaction par le gestionnaire OnTradeTransaction. Le changement de la position (l'ajout, le changement ou la	

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	liquidation) à la suite de l'exécution du marché n'entraîne pas l'apparition de la transaction TRADE_TRANSACTION_POSITION.	
TRADE_TRANSACTION_REQUEST	La notification que la demande commerciale est traitée par le serveur, et le résultat de son traitement est reçu. Pour les transaction de ce type dans la structureMqlTradeTransaction il faut analyser seulement un	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
	champ - type (le type de la transaction). Pour la réception de l'information supplémentaire il est nécessaire d'analyser les deuxièmes et troisièmes paramètres de la fonction OnTradeTransaction (request et result).	
TUESDAY	Mardi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TYPE_BOOL	bool	MqlParam
TYPE_CHAR	char	MqlParam
TYPE_COLOR	color	MqlParam
TYPE_DATETIME	datetime	MqlParam
TYPE_DOUBLE	double	MqlParam
TYPE_FLOAT	float	MqlParam
TYPE_INT	int	MqlParam

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
TYPE_LONG	long	MqlParam
TYPE_SHORT	short	MqlParam
TYPE_STRING	string	MqlParam
TYPE_UCHAR	uchar	MqlParam
TYPE_UINT	uint	MqlParam
TYPE_ULONG	ulong	MqlParam
TYPE_USHORT	ushort	MqlParam
UCHAR_MAX	La valeur maximale, qui peut être présentée par le type uchar	Constantes des types de données numériques
UINT_MAX	La valeur maximale, qui peut être présentée par le type uint	Constantes des types de données numériques
ULONG_MAX	La valeur maximale, qui peut être présentée par le type ulong	Constantes des types de données numériques
UPPER_BAND	Une limite supérieure	Lignes des indicateurs

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
UPPER_HISTOGRAM	L'histogramme supérieur	Lignes des indicateurs
UPPER_LINE	Une ligne supérieure	Lignes des indicateurs
USHORT_MAX	La valeur maximale, qui peut être présentée par le type ushort	Constantes des types de données numériques
VOLUME_REAL	Le volume commercial	Constantes de prix
VOLUME_TICK	Le volume de tick	Constantes de prix
WEDNESDAY	Mercredi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
WHOLE_ARRAY	Signifie le nombre d'éléments restés jusqu'à la fin du tableau, c'est-à-dire, tout le tableau sera traité	Les autres constantes

© 2000-2024, MetaQuotes Ltd.

Constante	Description	Использование
WRONG_VALUE	La constante peut non évidemment être amenée au type de n'importe quelle énumération	Les autres constantes

© 2000-2024, MetaQuotes Ltd.